Modern Software Engineering

Building Better Software Faster

Dave Farley

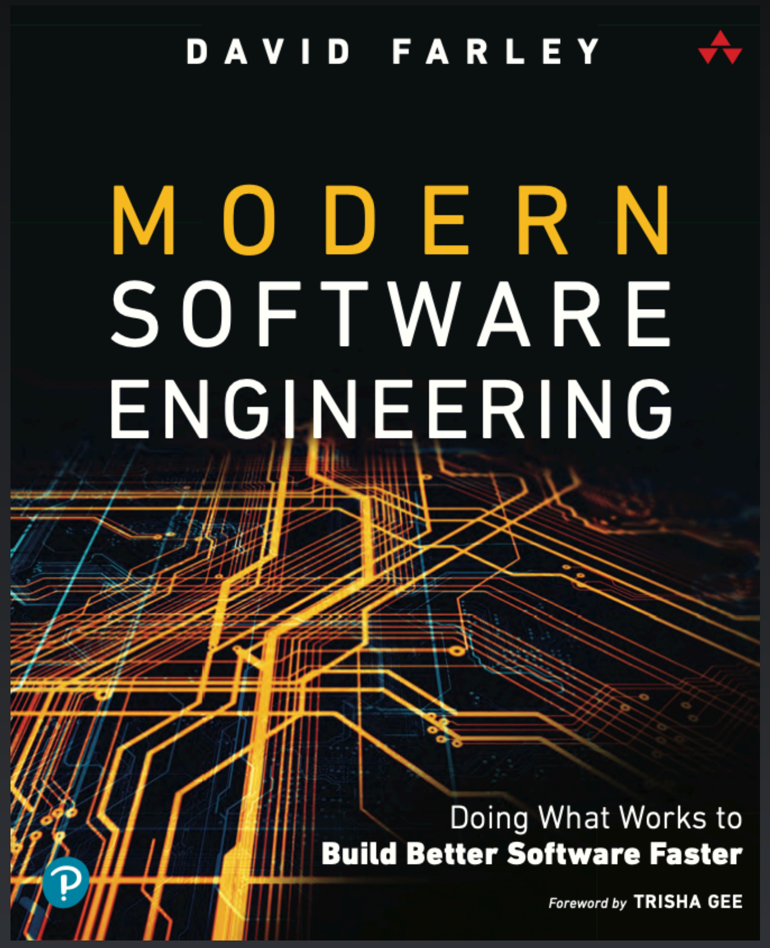
https://www.davefarley.net

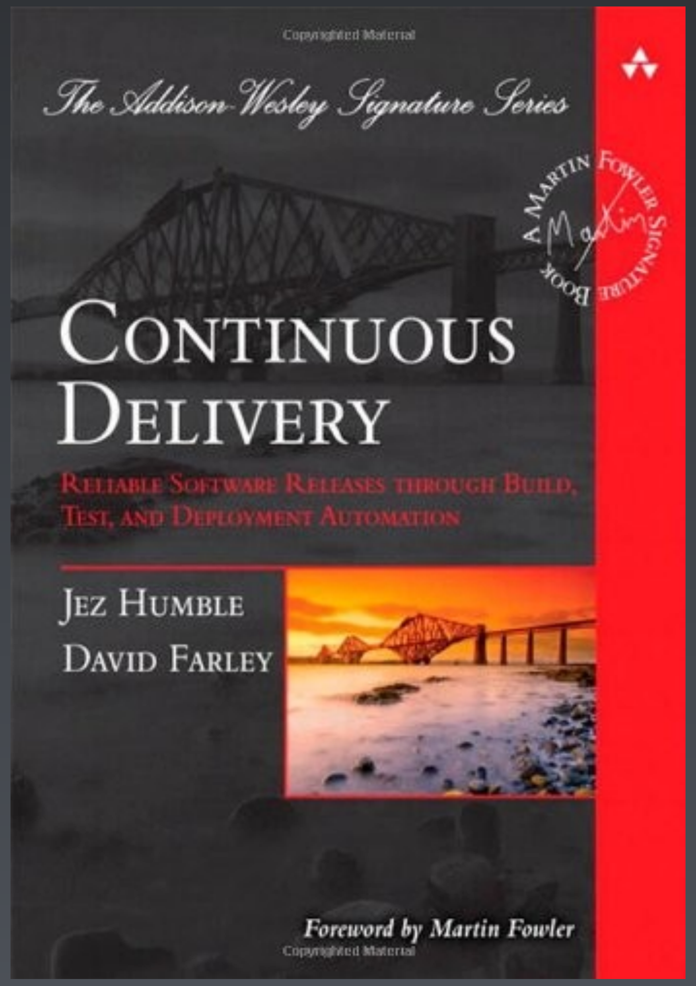
@davefarley77

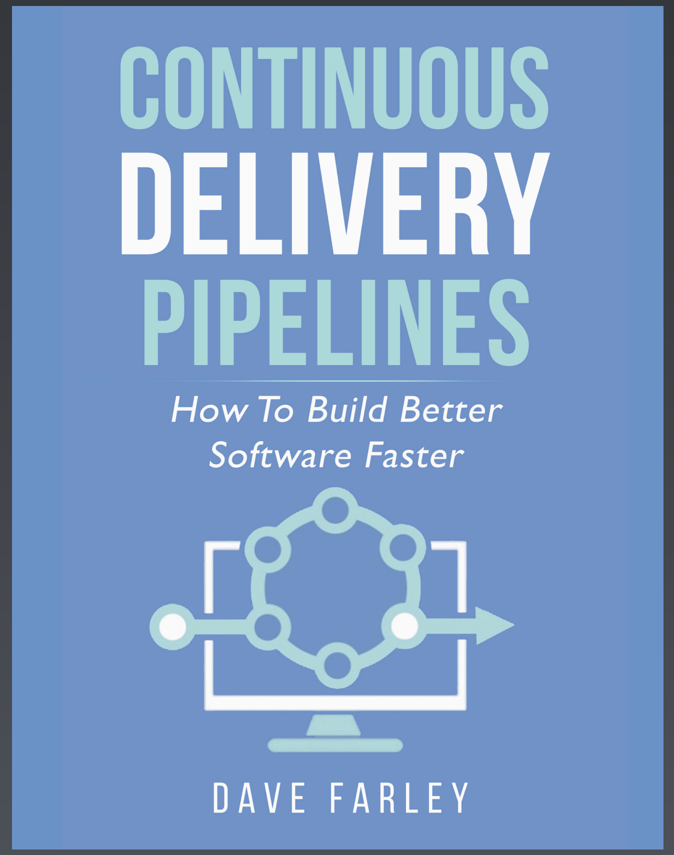
https://bit.ly/CDonYT

Continuous Delivery ltd

http://www.continuous-delivery.co.uk


DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to Build Better Software Faster
Foreword by TRISHA GEE


The Addison-Wesley Signature Series
Continuous Delivery
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler


CONTINUOUS DELIVERY PIPELINES
How To Build Better Software Faster
DAVE FARLEY

- ***What is Engineering***

Continuous
Delivery ltd

- ***What is Engineering***
- ***Common Foundational Principles***

- *What is Engineering*
- *Common Foundational Principles*
- *10 Guiding Principles for Software Engineering*

- What is Engineering
- Common Foundational Principles
- 10 Guiding Principles for Software Engineering
- Applying the Guidelines to Code

What is Engineering

Continuous Delivery ltd

- ***Engineering in Other Disciplines is:***

  "The Stuff That Works"

- Engineering in Other Disciplines is:
  "The Stuff That Works"

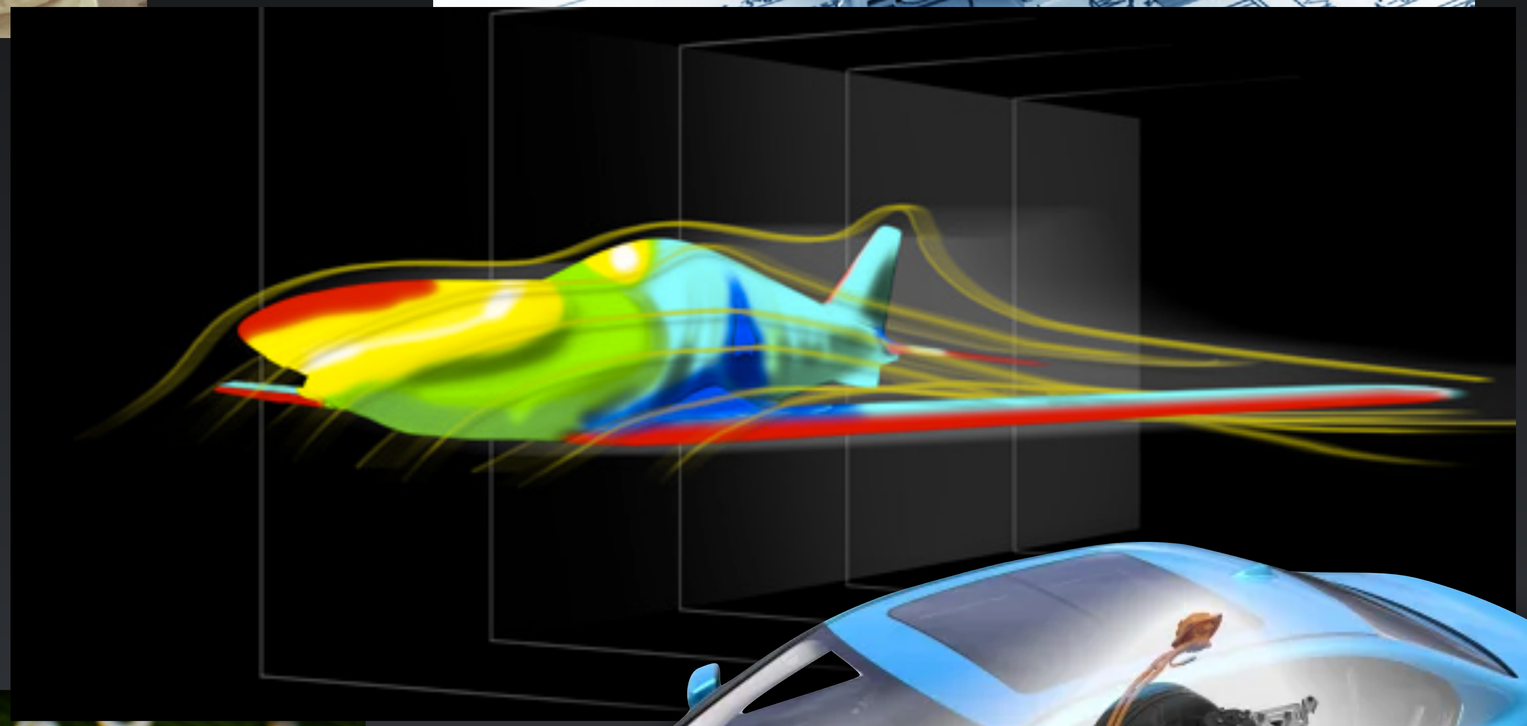
- Software is Difficult!
  So What "Works" for Us?

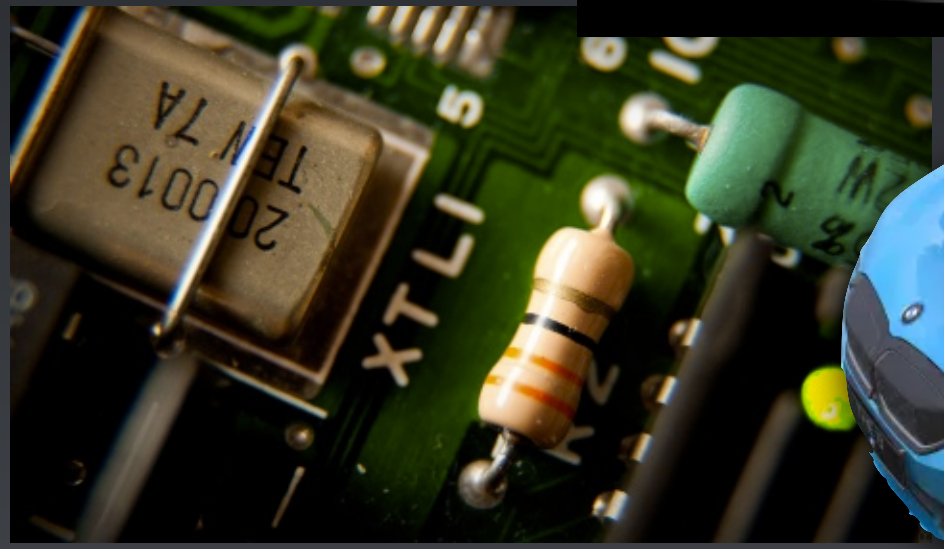
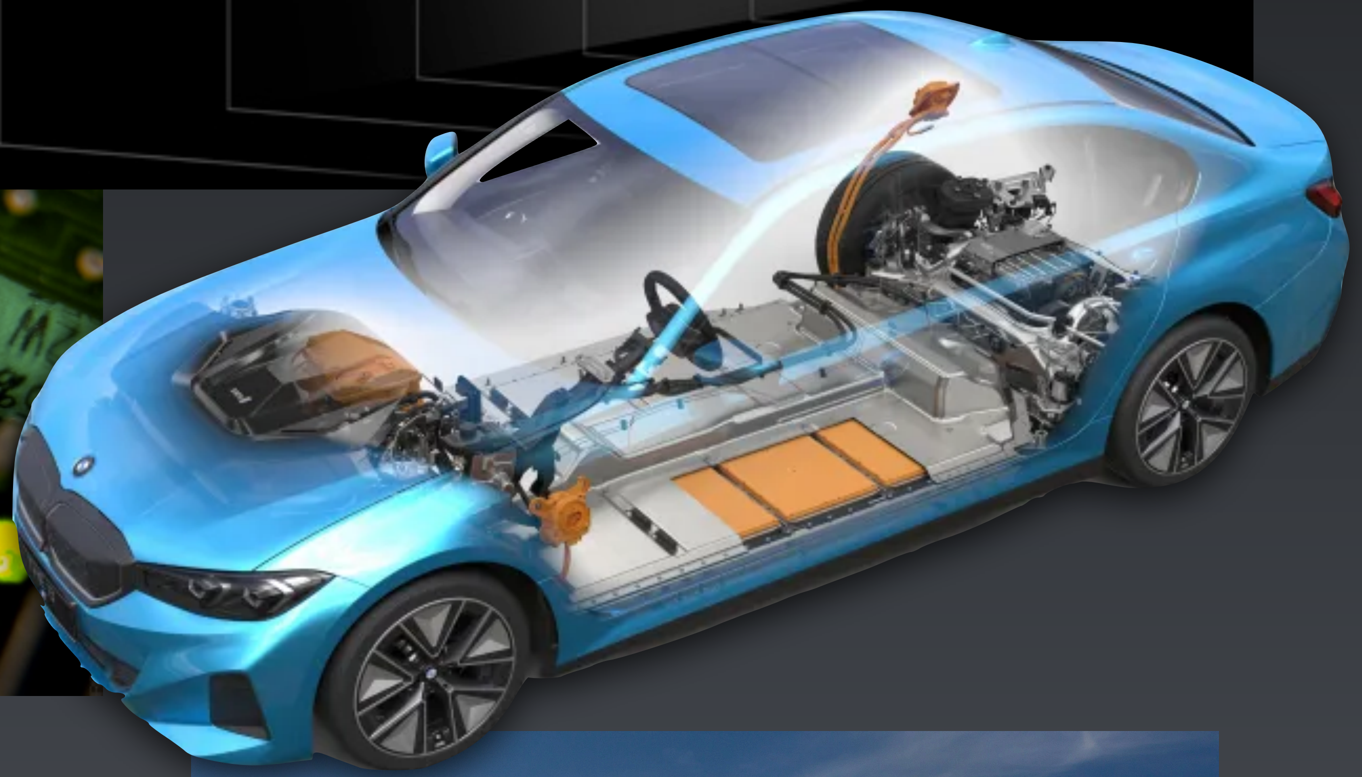
Continuous Delivery ltd

All Engineering is not the same!

Continuous Delivery ltd

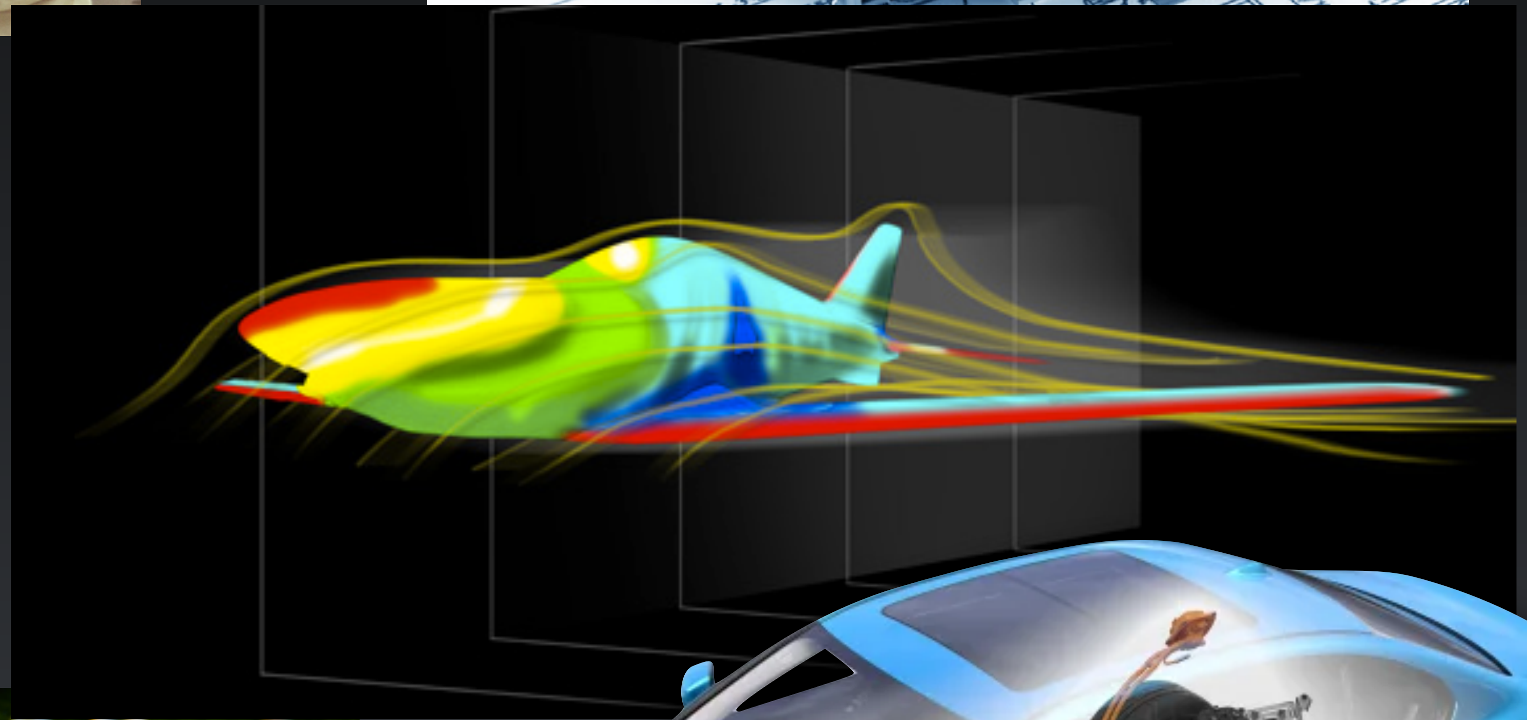
All Engineering is not the same!

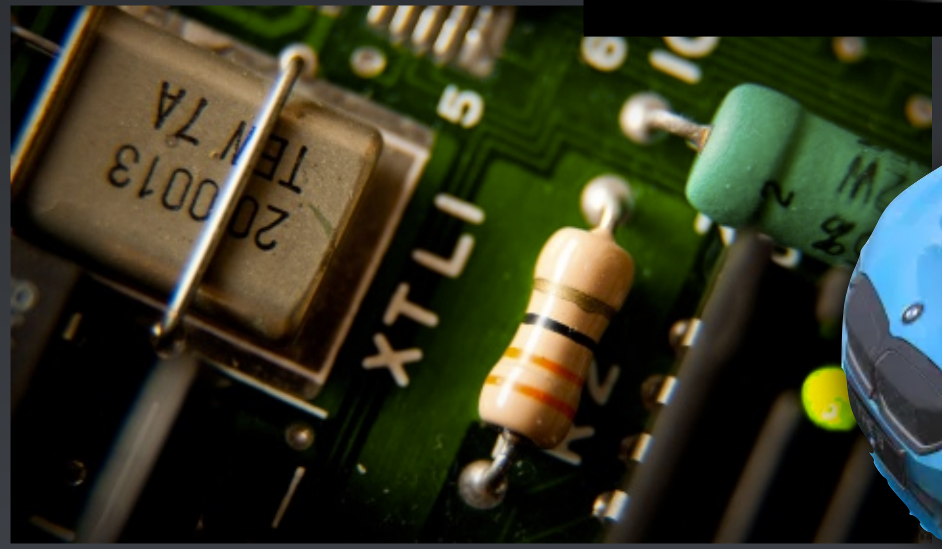
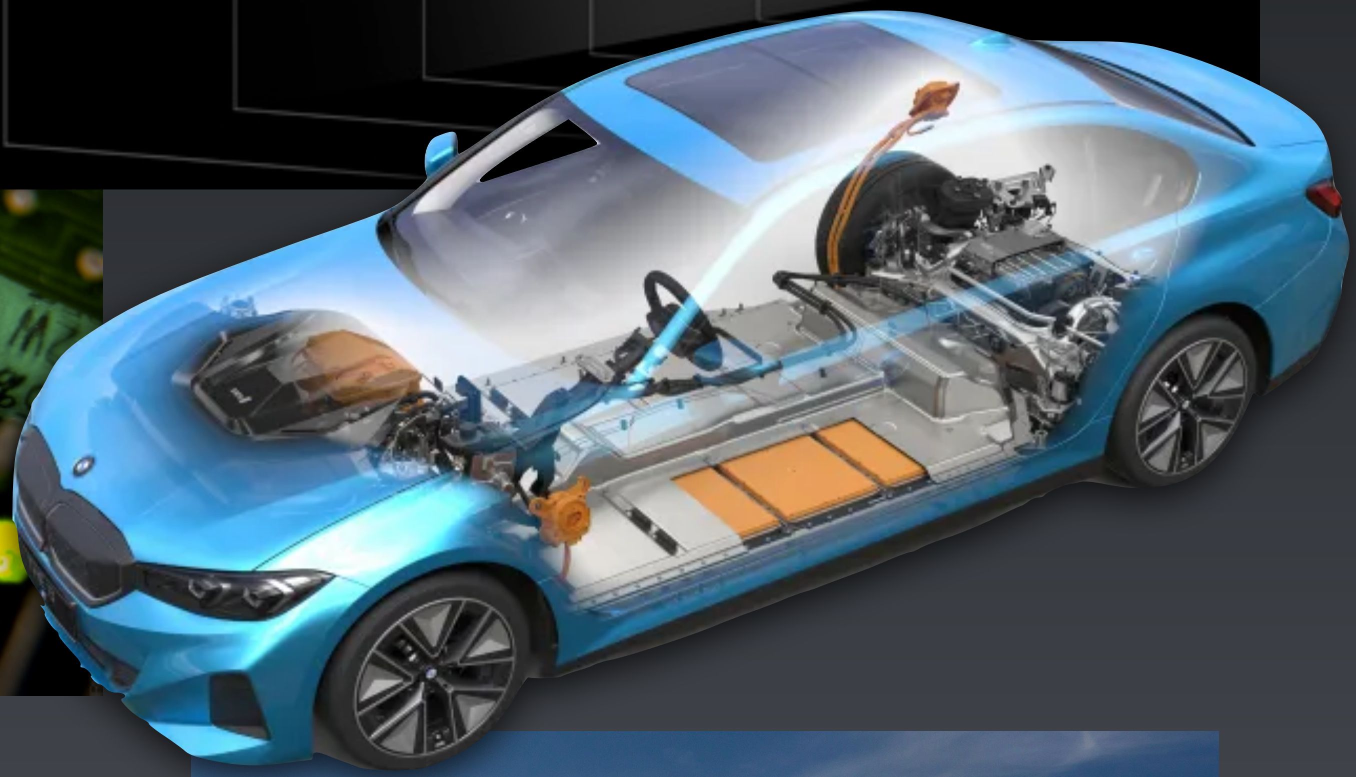


Continuous
Delivery ltd

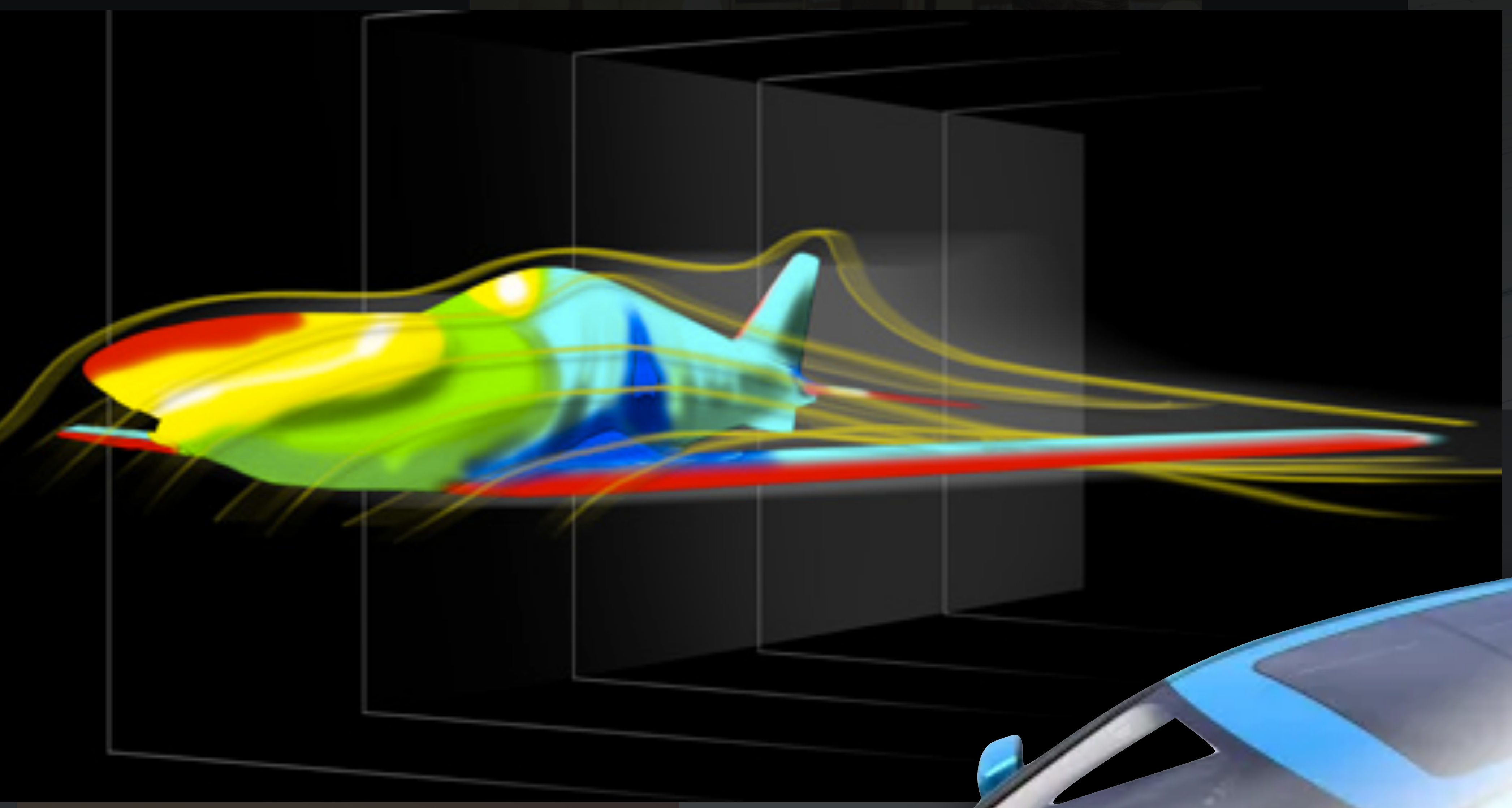
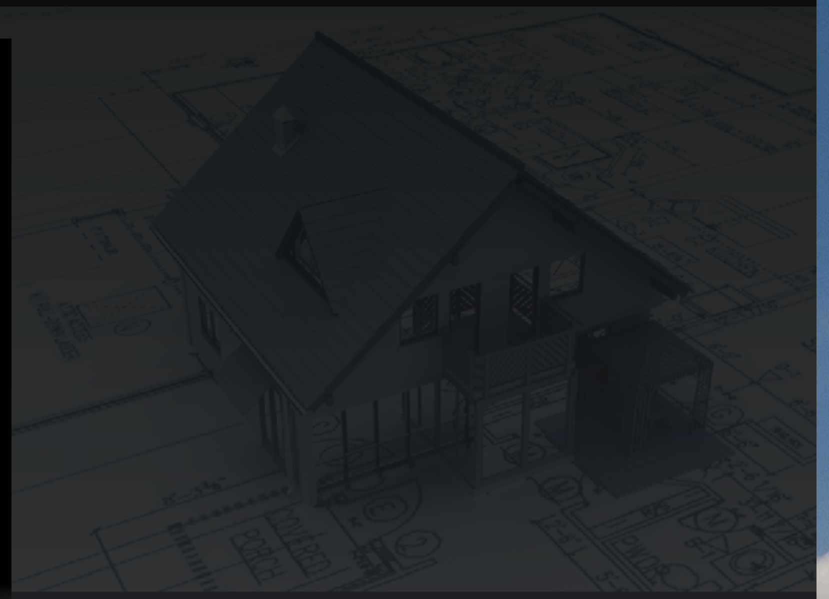
Learning From Other Engineering Disciplines

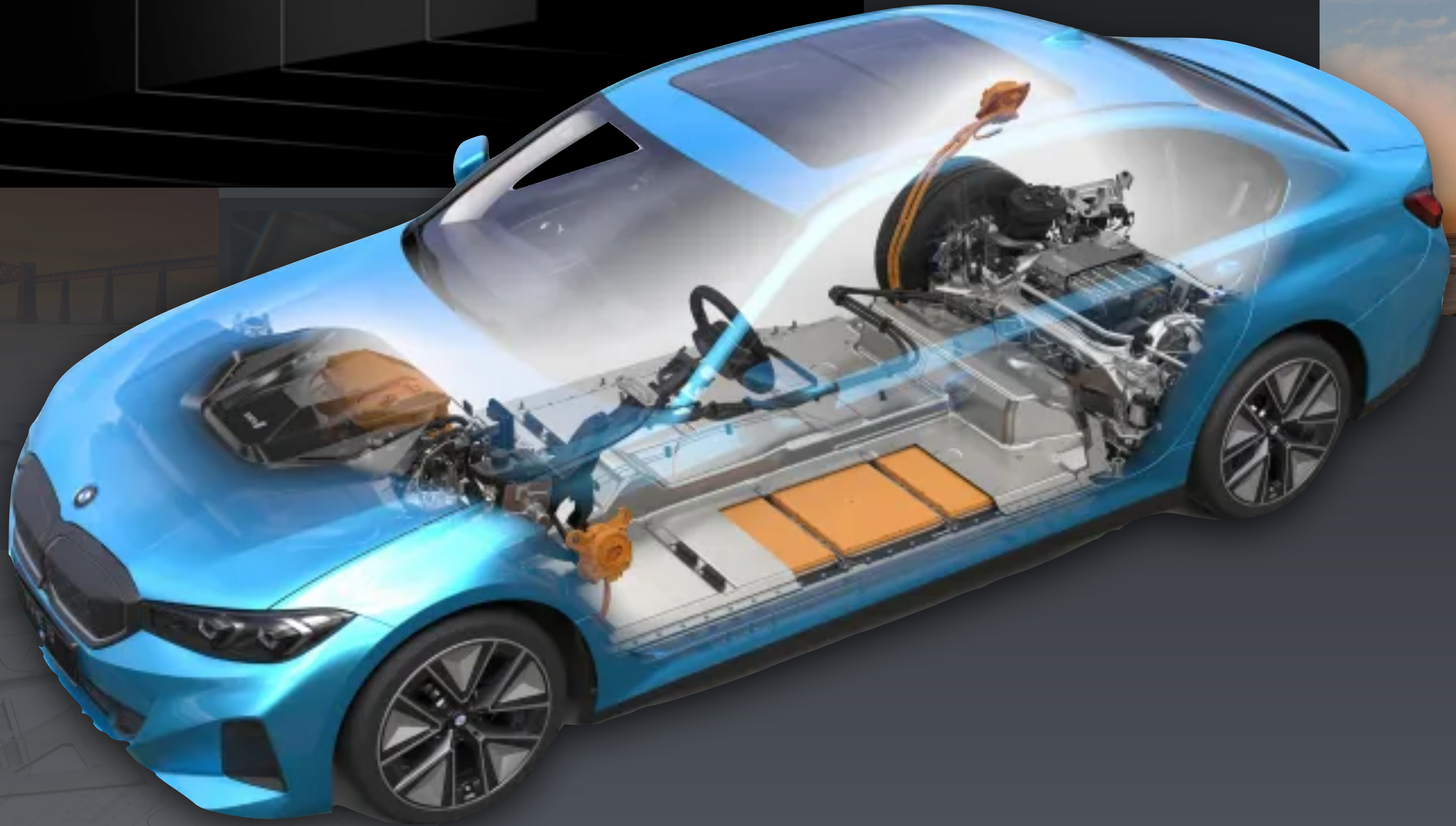


Continuous Delivery ltd

Learning From Other Engineering Disciplines



Continuous Delivery ltd

Learning From Other Engineering Disciplines

Continuous Delivery ltd

Learning From Other Engineering Disciplines

What Does This Mean for Us?

Continuous Delivery ltd

What is ‘Engineering’?

Continuous Delivery ltd

What is ‘Engineering’?

***Engineering** is the application of an empirical, scientific approach to finding efficient solutions to practical problems.*

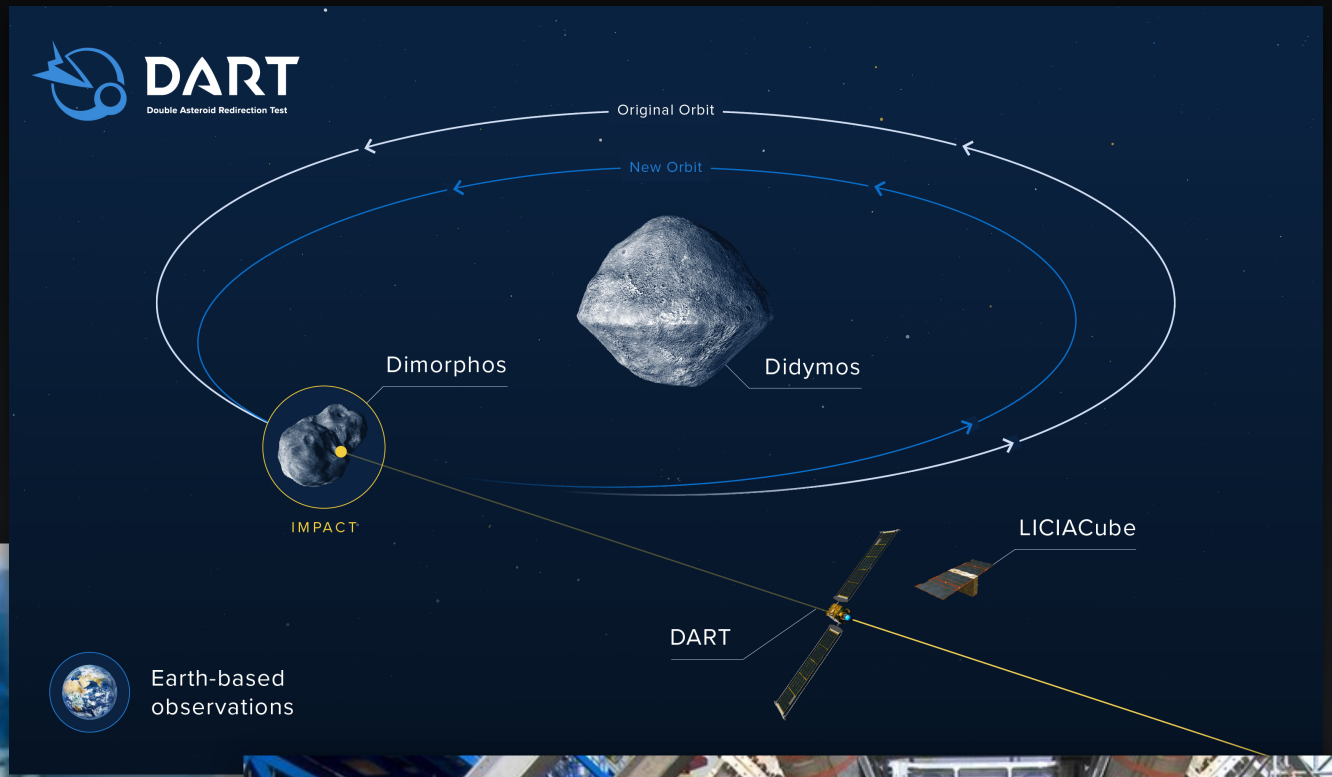
(Dave Farley - Just Now!)

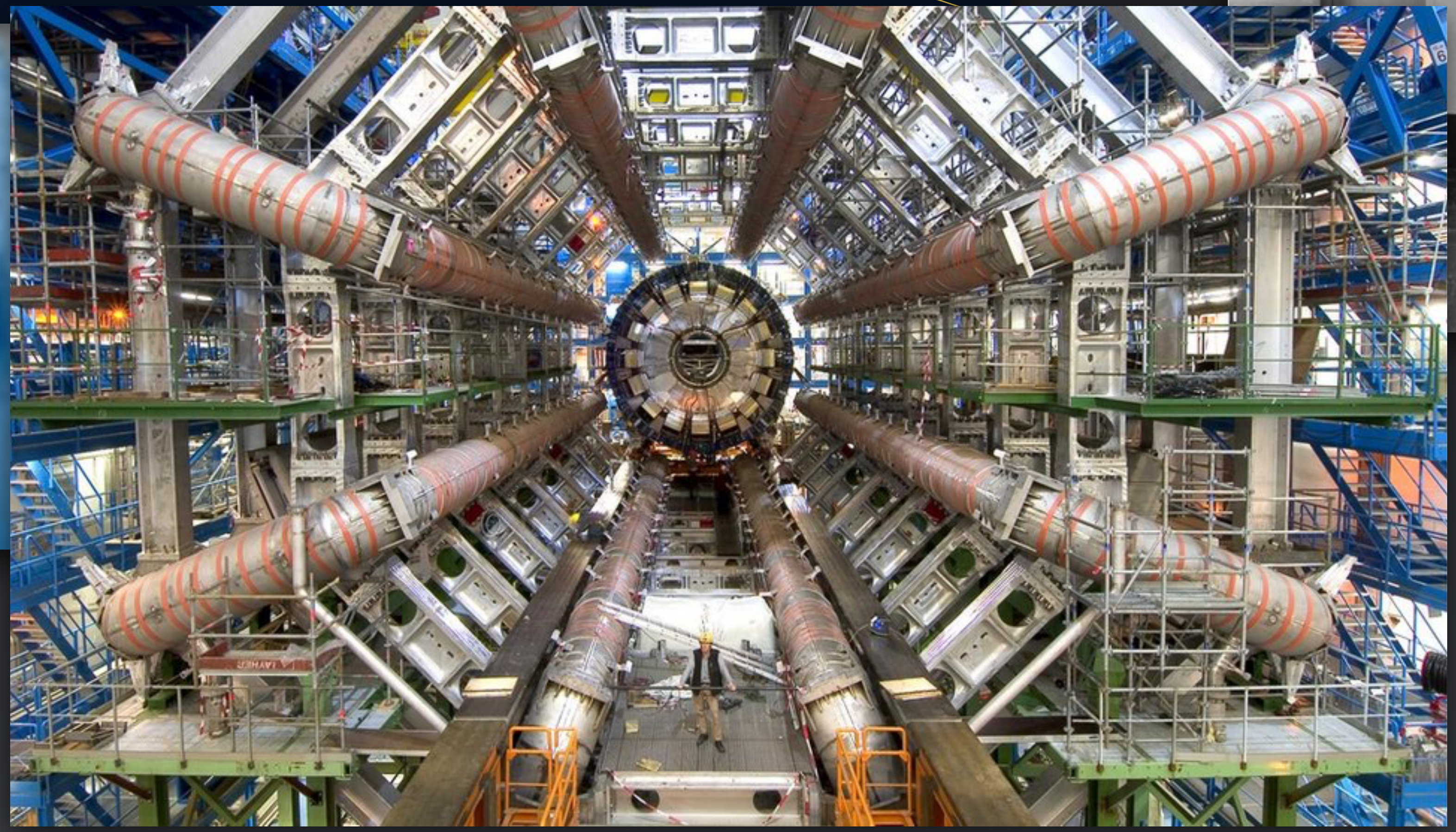
Continuous Delivery ltd

To Put It More Simply…

Continuous Delivery ltd

To Put It More Simply…





Engineering

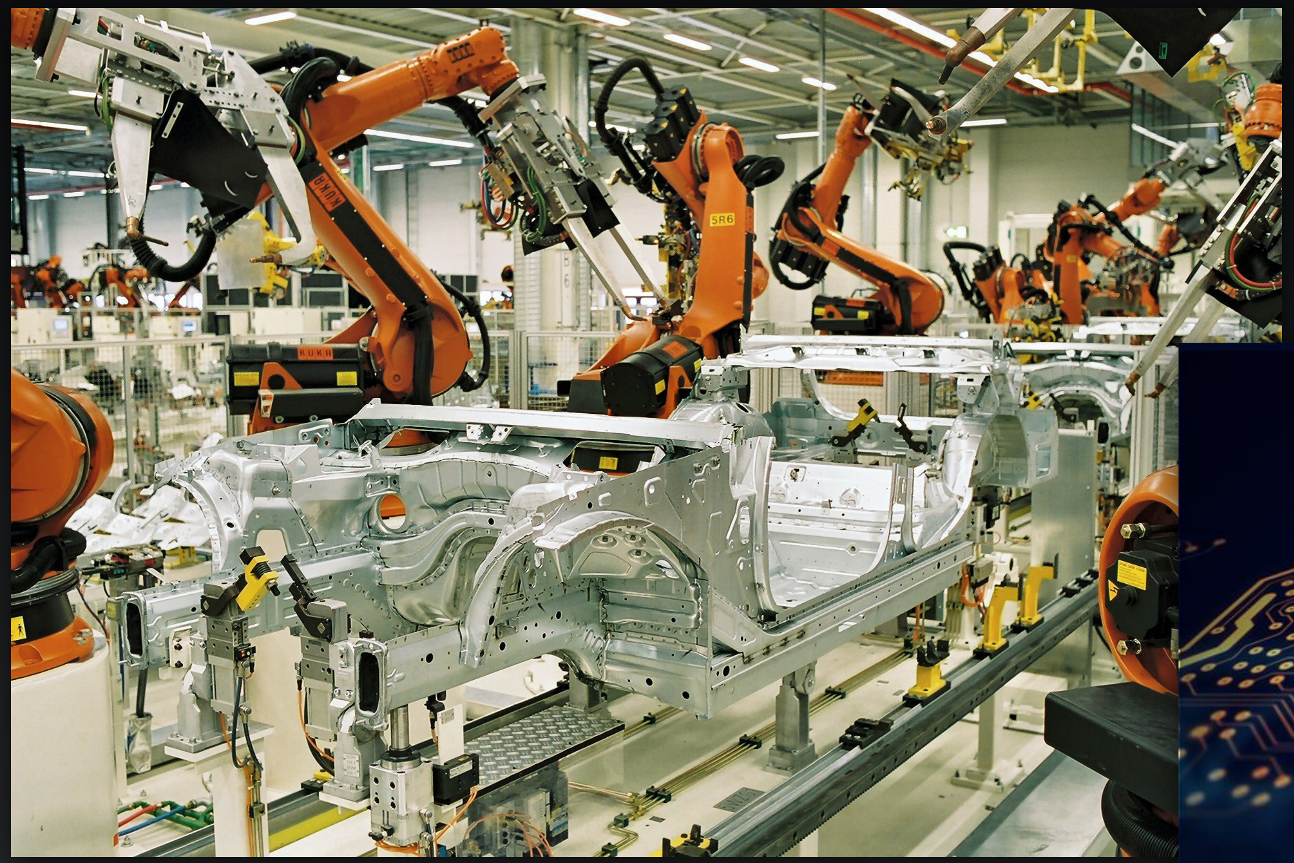
is How We Solve the Hard Problems!

Continuous Delivery ltd

To Put It More Simply…

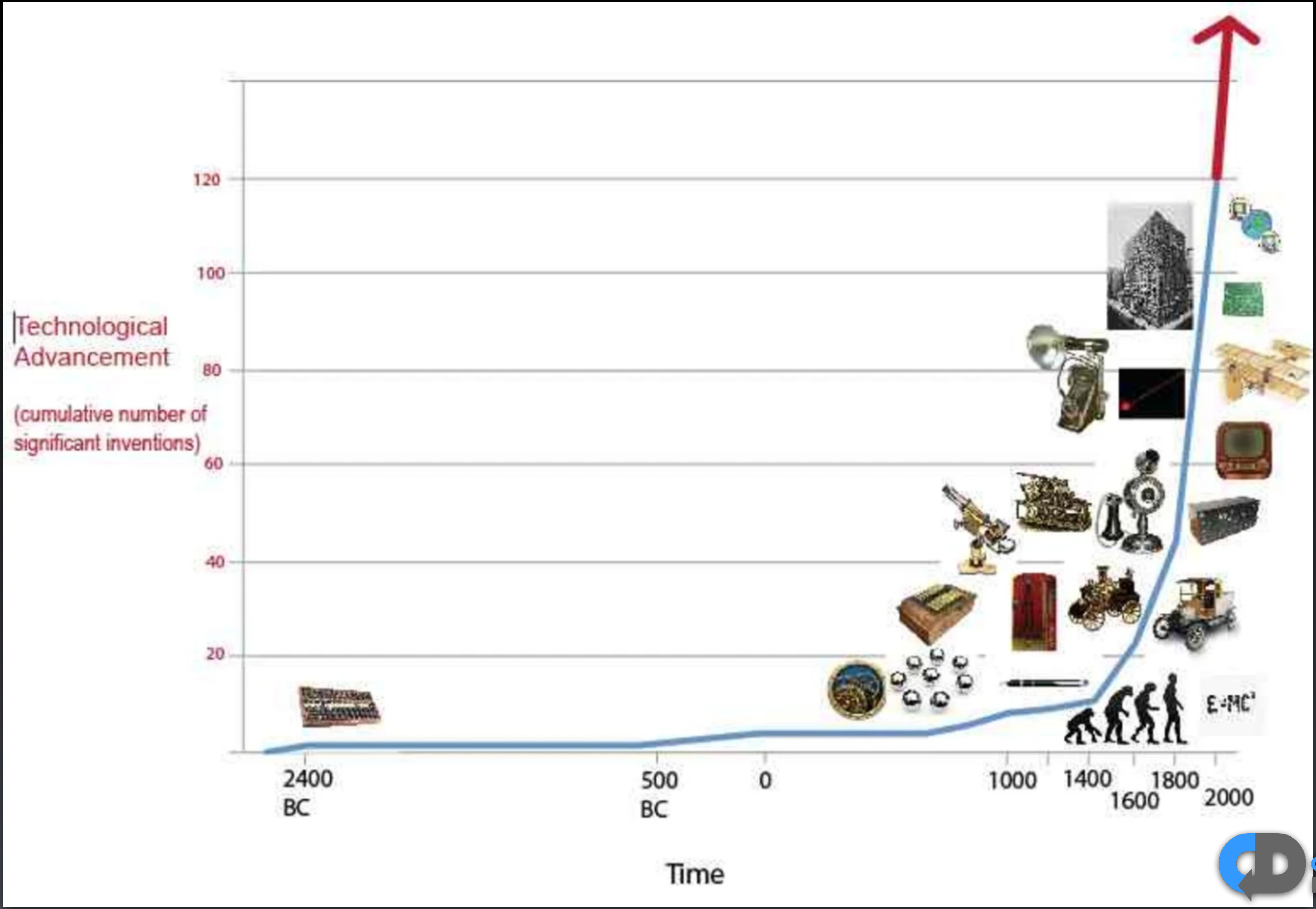
Continuous Delivery ltd

To Put It More Simply…

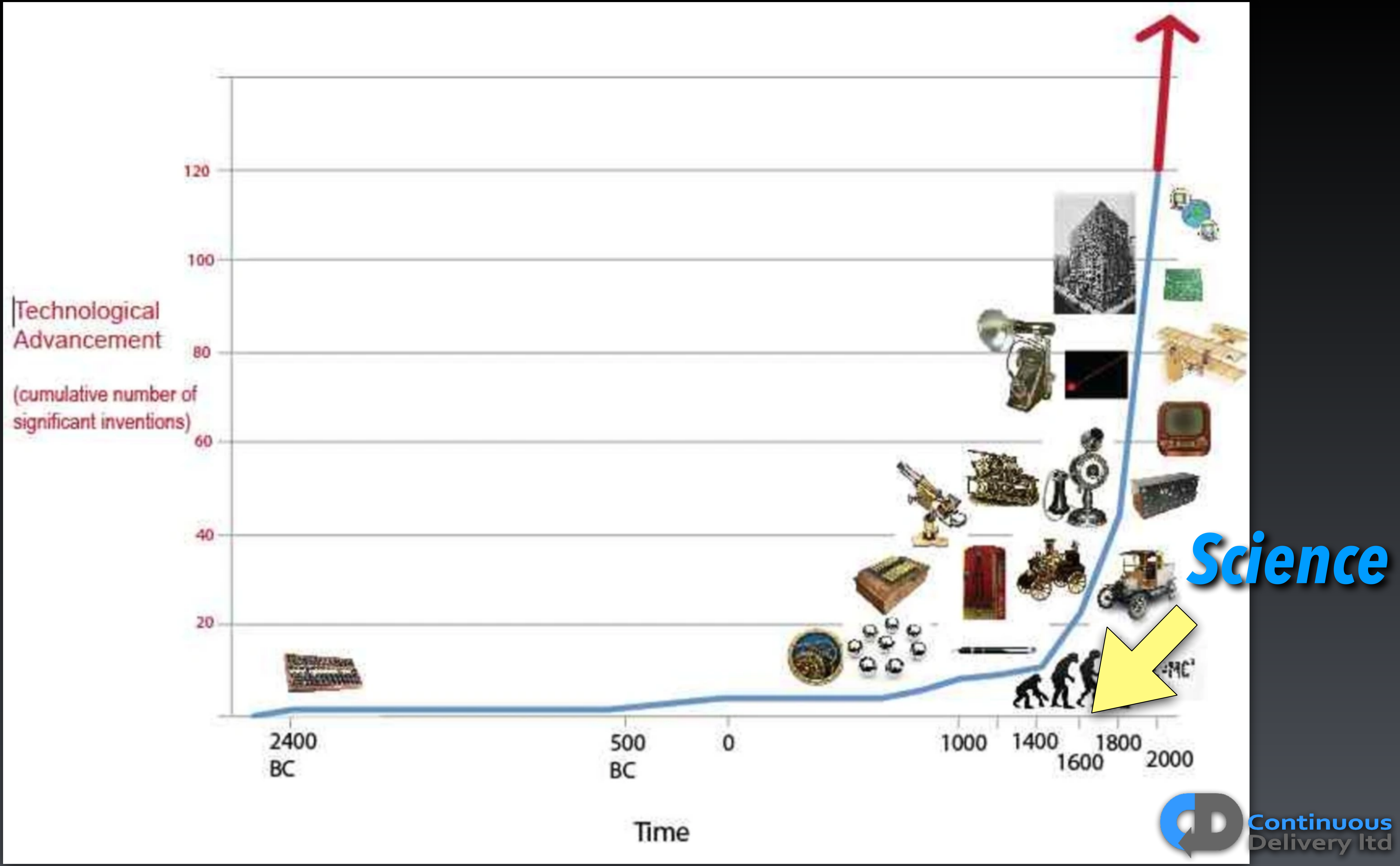


Engineering

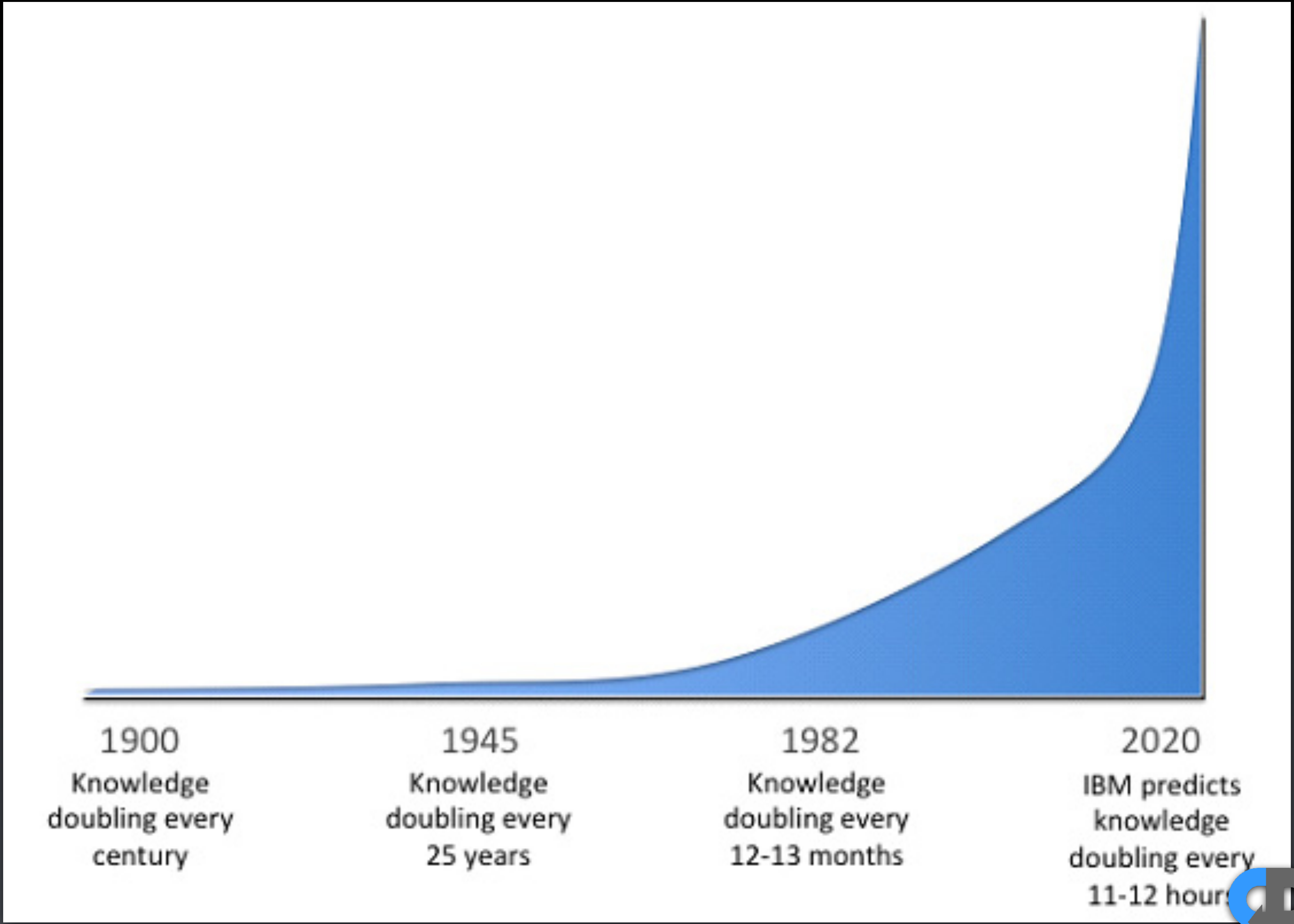
The Foundation on Which Our High-Tech Culture is Built!



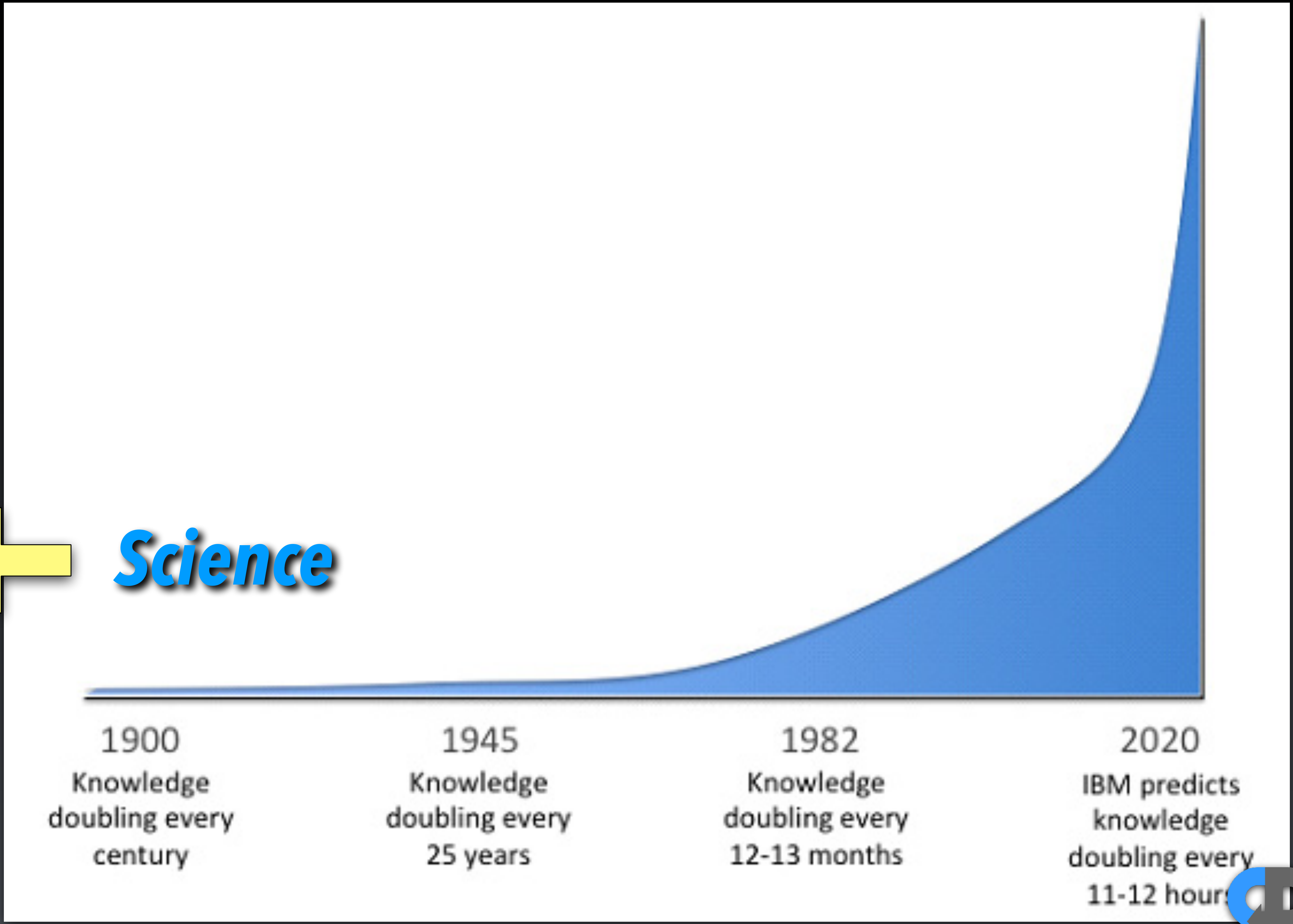
Continuous Delivery ltd



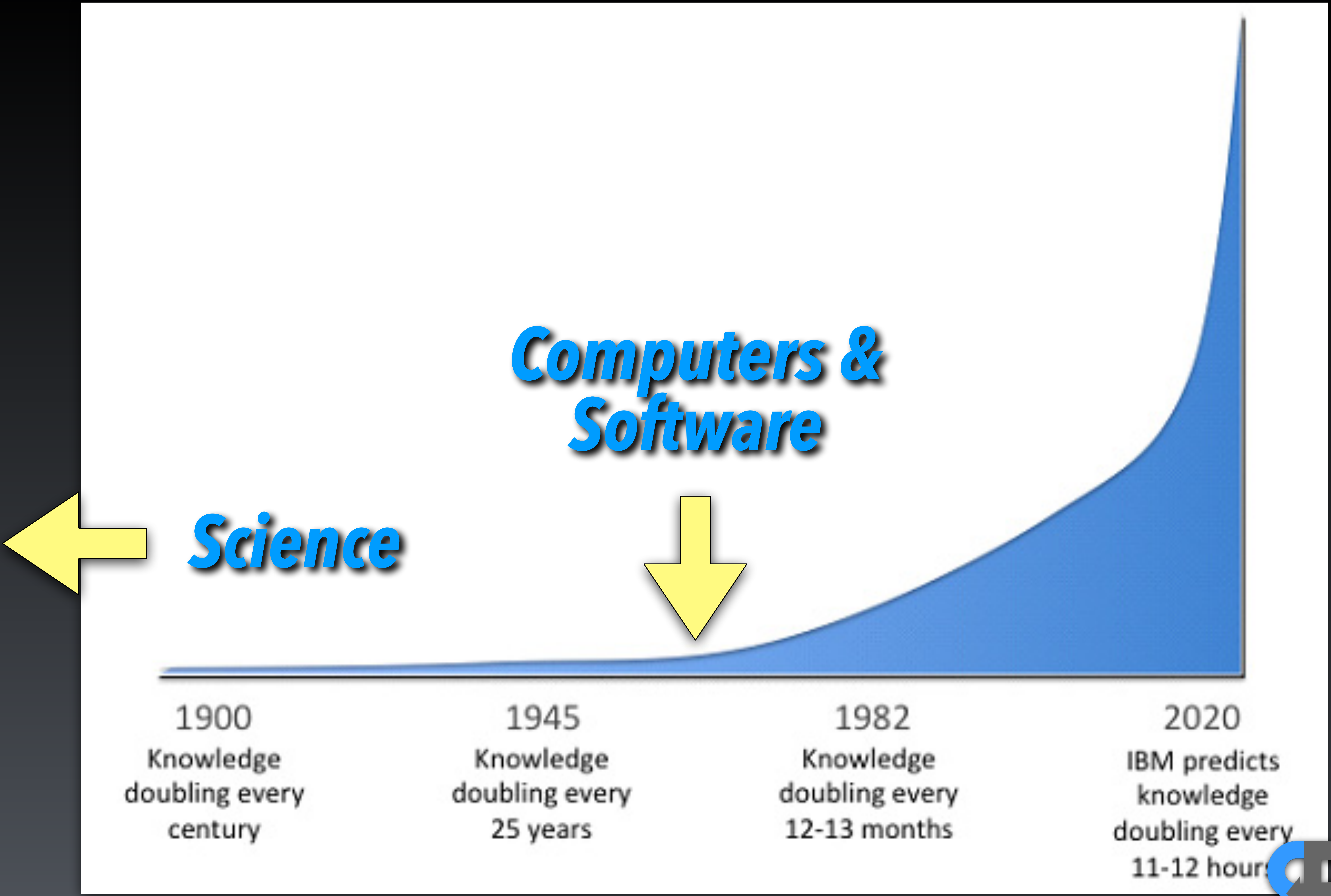
Continuous Delivery ltd



Continuous Delivery ltd



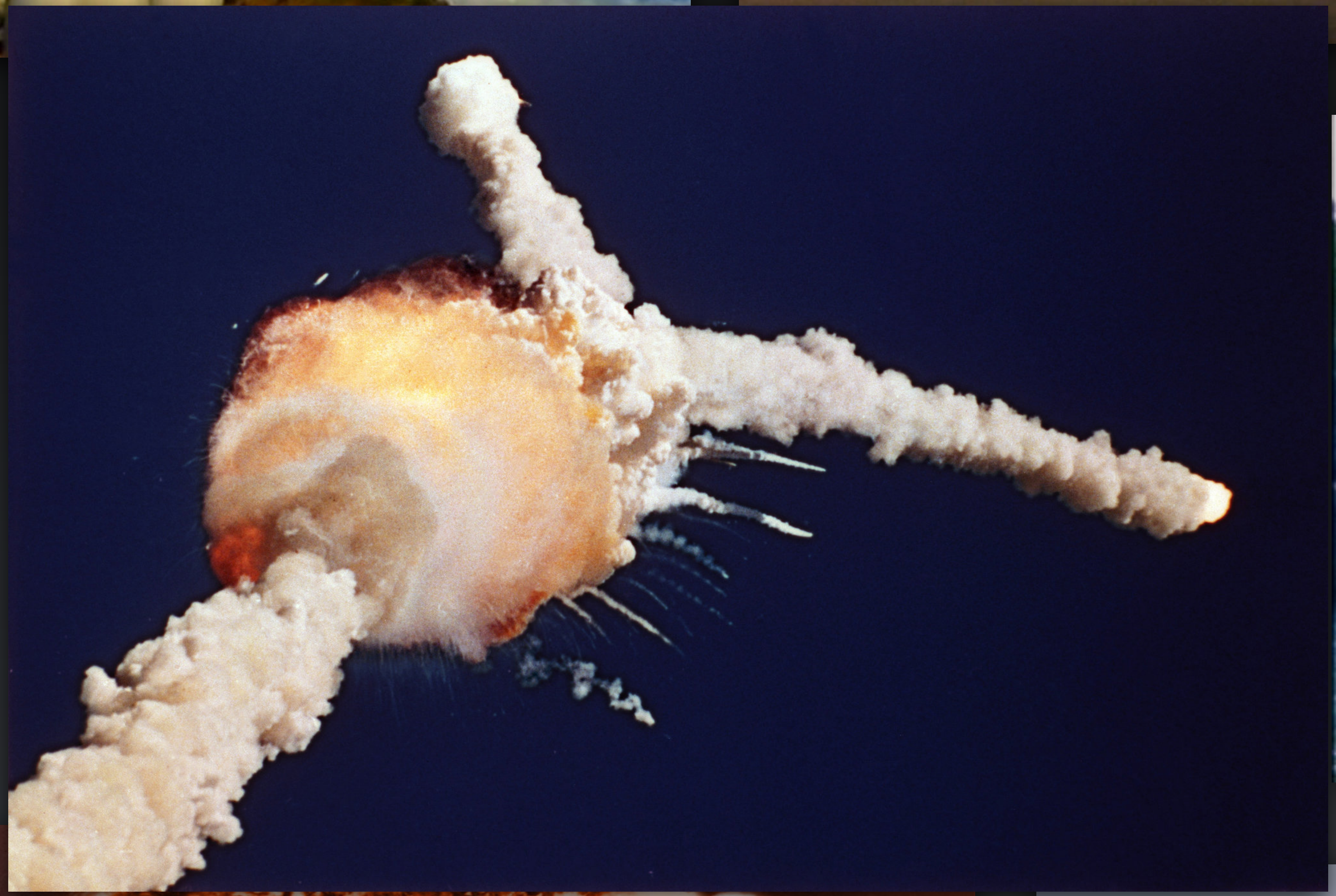
Continuous Delivery ltd



Continuous Delivery ltd

What is Engineering Really For?

Continuous Delivery ltd



Continuous Delivery ltd

It's Not to Define Solutions!

Continuous Delivery ltd

It Rules Out Bad Ideas!

Continuous Delivery ltd

“The Things We Can’t Afford to Get Wrong”

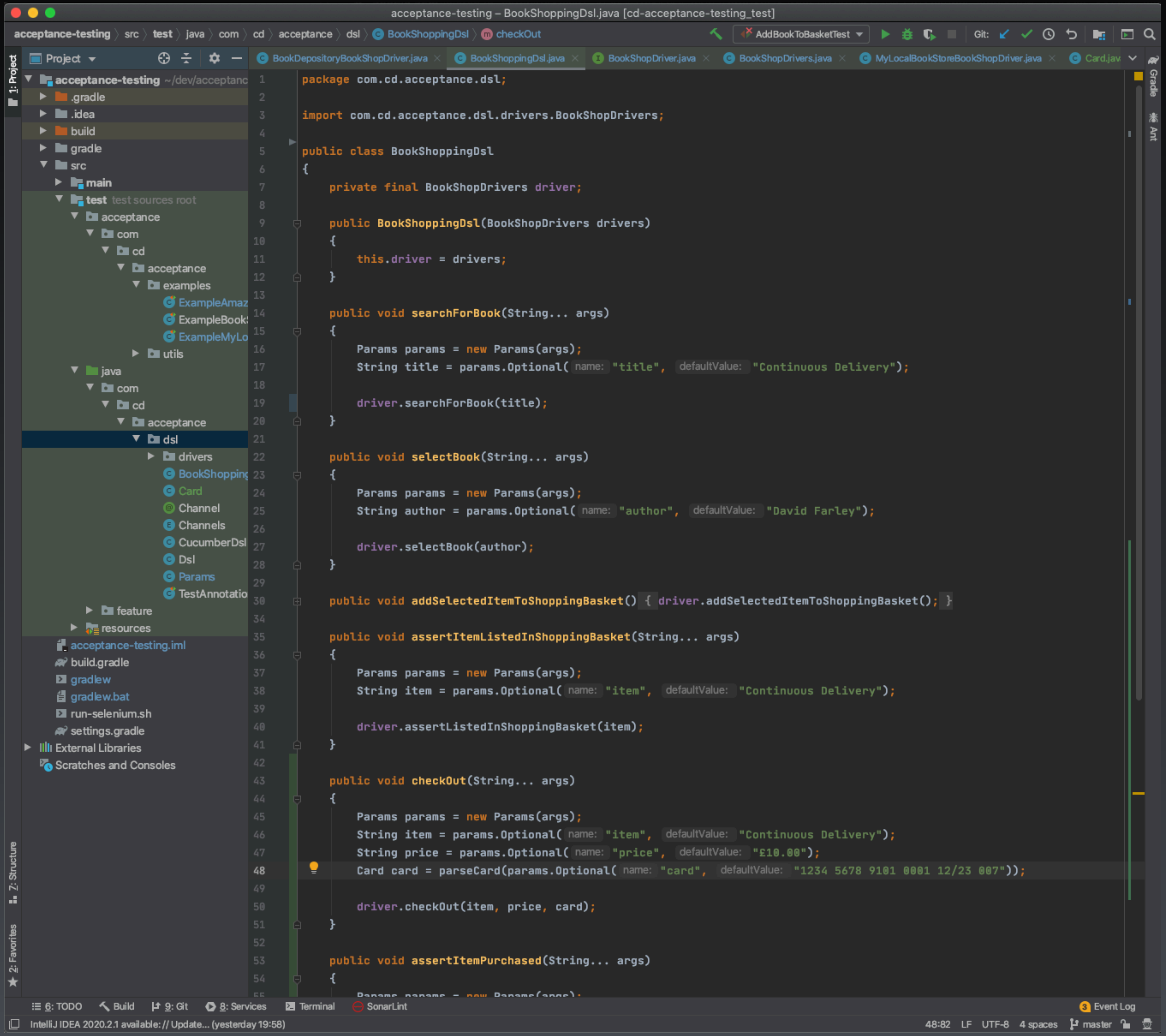
“The Things We Can’t Afford to Get Wrong”

Continuous Delivery ltd

~~“The Things We Can’t Afford to Get Wrong”~~

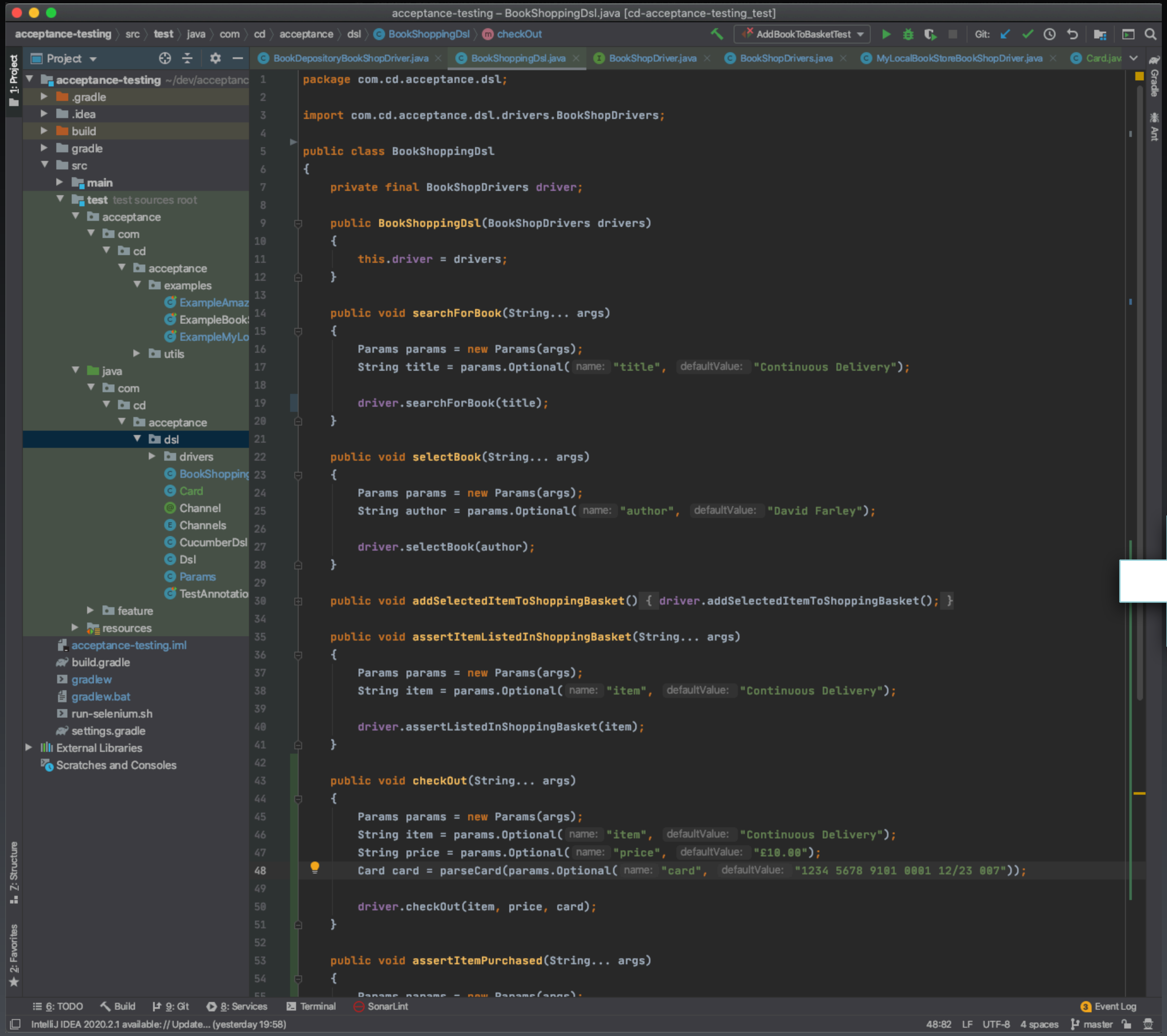
This is NOT Our Problem!

Continuous Delivery ltd

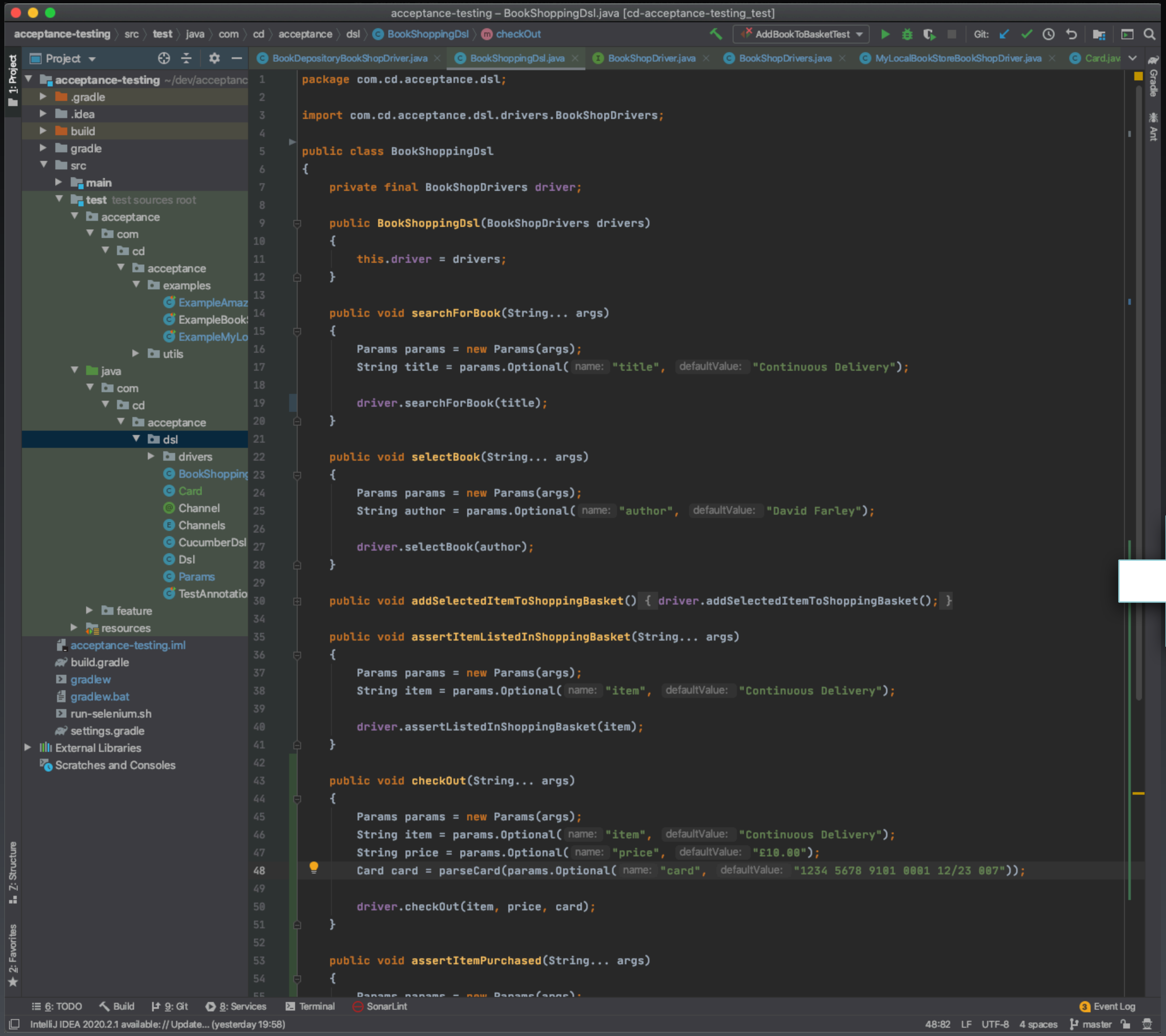


```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional( name: "title",  defaultValue: "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional( name: "author",  defaultValue: "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price",  defaultValue: "£10.00");
        Card card = parseCard(params.Optional( name: "card",  defaultValue: "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
        Params params = new Params(args);
```

Continuous Delivery ltd


acceptance-testing – BookShoppingDsl.java [cd-acceptance-testing_test]
BookDepositoryBookShopDriver.java
BookShoppingDsl.java
BookShopDriver.java
BookShopDrivers.java
MyLocalBookStoreBookShopDriver.java
AddBookToBasketTest

```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional( name: "title",  defaultValue: "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional( name: "author",  defaultValue: "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price",  defaultValue: "£10.00");
        Card card = parseCard(params.Optional( name: "card",  defaultValue: "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
        Params params = new Params(args);
```



```java
package com.cd.acceptance.dsl;

import com.cd.acceptance.dsl.drivers.BookShopDrivers;

public class BookShoppingDsl
{
    private final BookShopDrivers driver;

    public BookShoppingDsl(BookShopDrivers drivers)
    {
        this.driver = drivers;
    }

    public void searchForBook(String... args)
    {
        Params params = new Params(args);
        String title = params.Optional( name: "title",  defaultValue: "Continuous Delivery");

        driver.searchForBook(title);
    }

    public void selectBook(String... args)
    {
        Params params = new Params(args);
        String author = params.Optional( name: "author",  defaultValue: "David Farley");

        driver.selectBook(author);
    }

    public void addSelectedItemToShoppingBasket() { driver.addSelectedItemToShoppingBasket(); }

    public void assertItemListedInShoppingBasket(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");

        driver.assertListedInShoppingBasket(item);
    }

    public void checkOut(String... args)
    {
        Params params = new Params(args);
        String item = params.Optional( name: "item",  defaultValue: "Continuous Delivery");
        String price = params.Optional( name: "price",  defaultValue: "£10.00");
        Card card = parseCard(params.Optional( name: "card",  defaultValue: "1234 5678 9101 0001 12/23 007"));

        driver.checkOut(item, price, card);
    }

    public void assertItemPurchased(String... args)
    {
```

Continuous Delivery ltd

DEPLOY

We Can Recreate Entire Systems for Free!

Continuous Delivery ltd

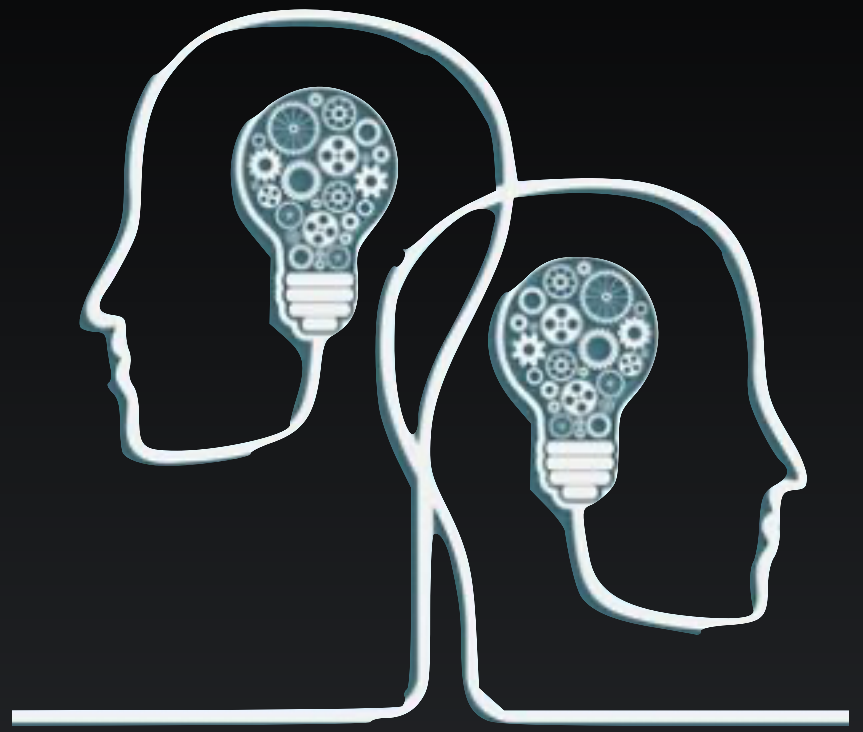
Engineering *IS* About Exploration & Discovery

Software Development *IS* About Exploration & Discovery

Continuous Delivery ltd

The Stuff That Works

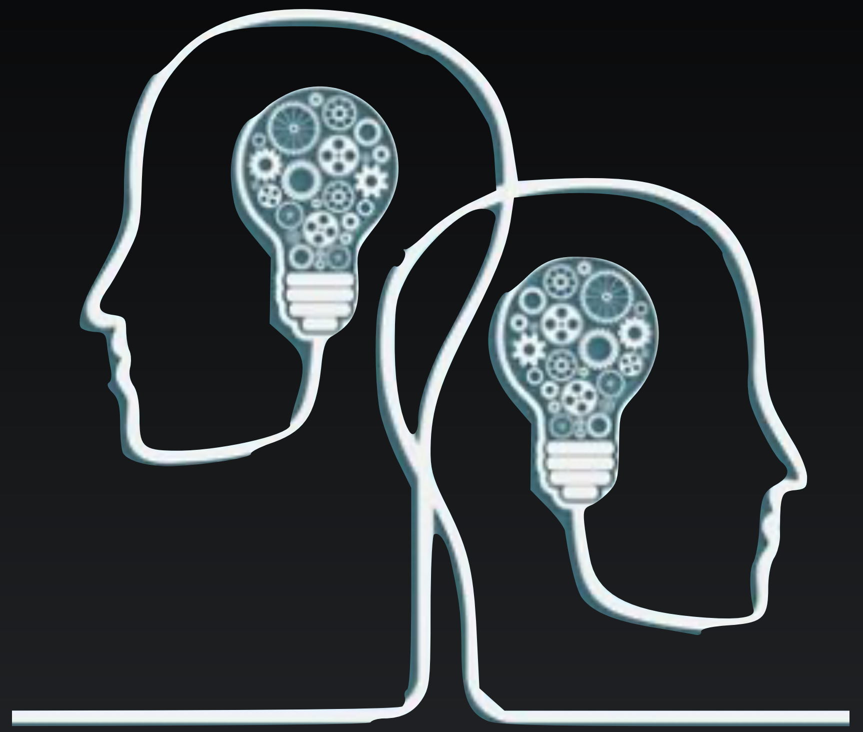
The Stuff That Works



Optimise for Learning

Continuous Delivery ltd

The Stuff That Works



Optimise for Learning

Optimise for Managing Complexity

Continuous Delivery ltd

Optimise for Learning

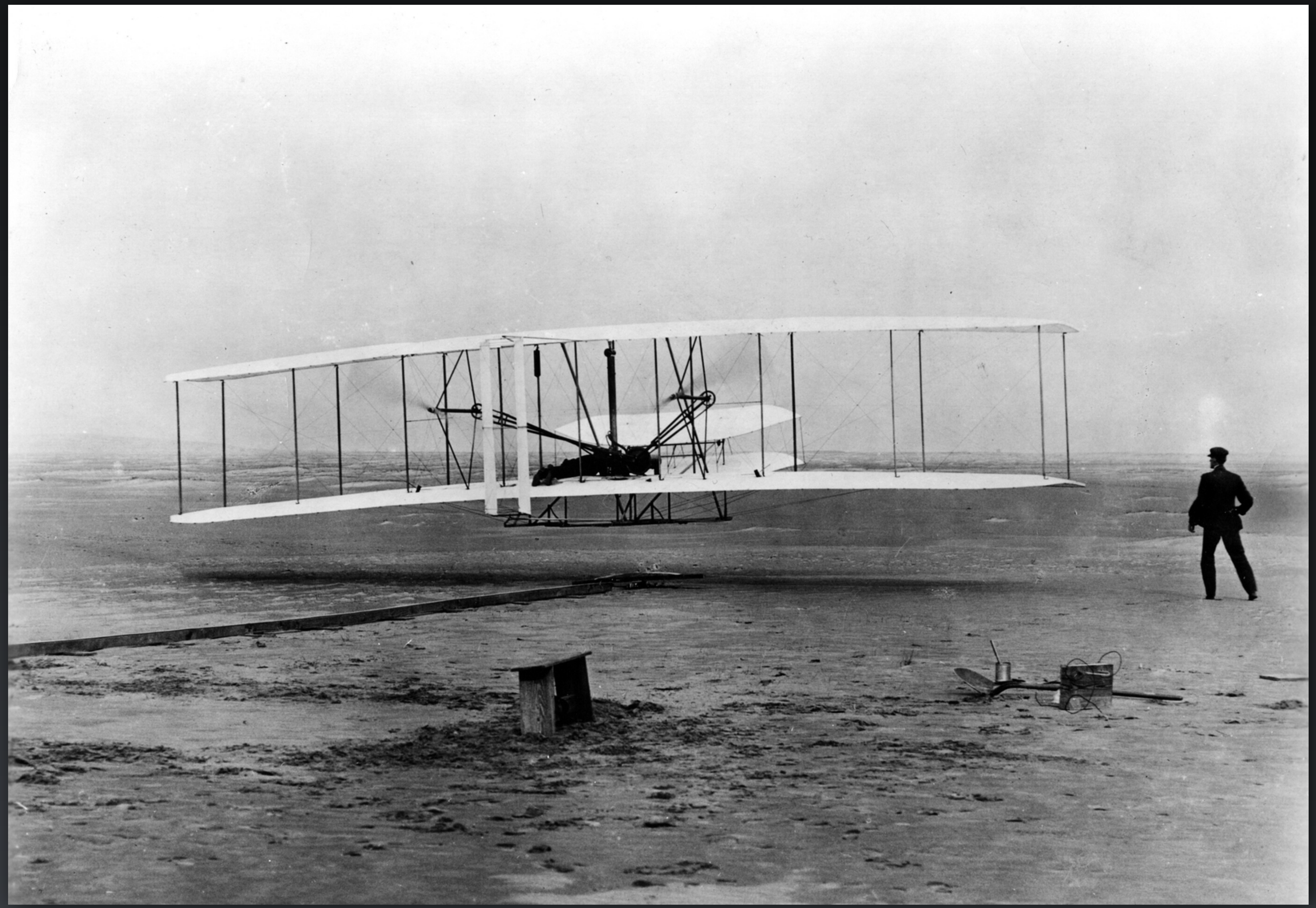
Iterative Experimental

Feedback Driven

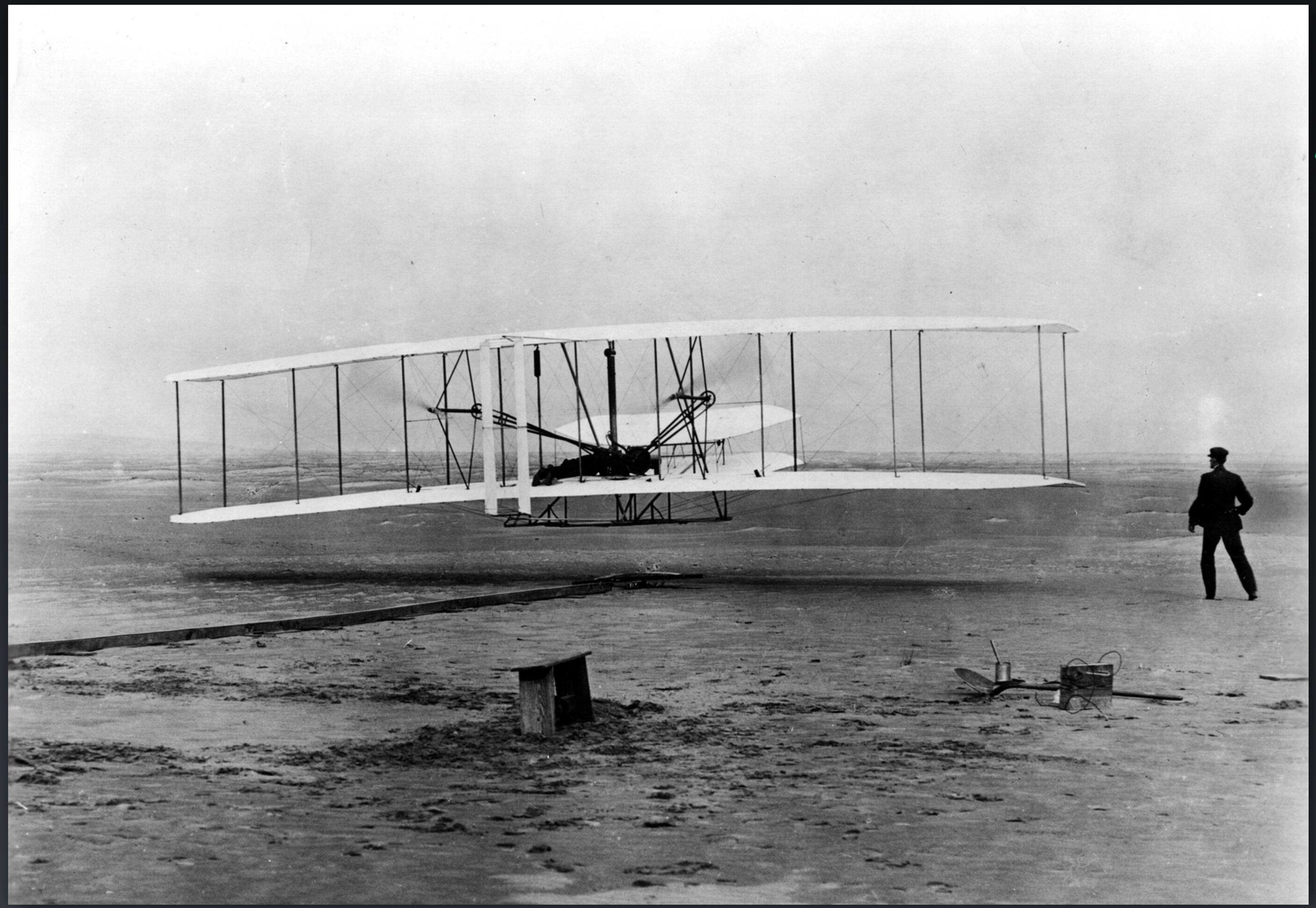
Incremental Empirical

Continuous Delivery ltd

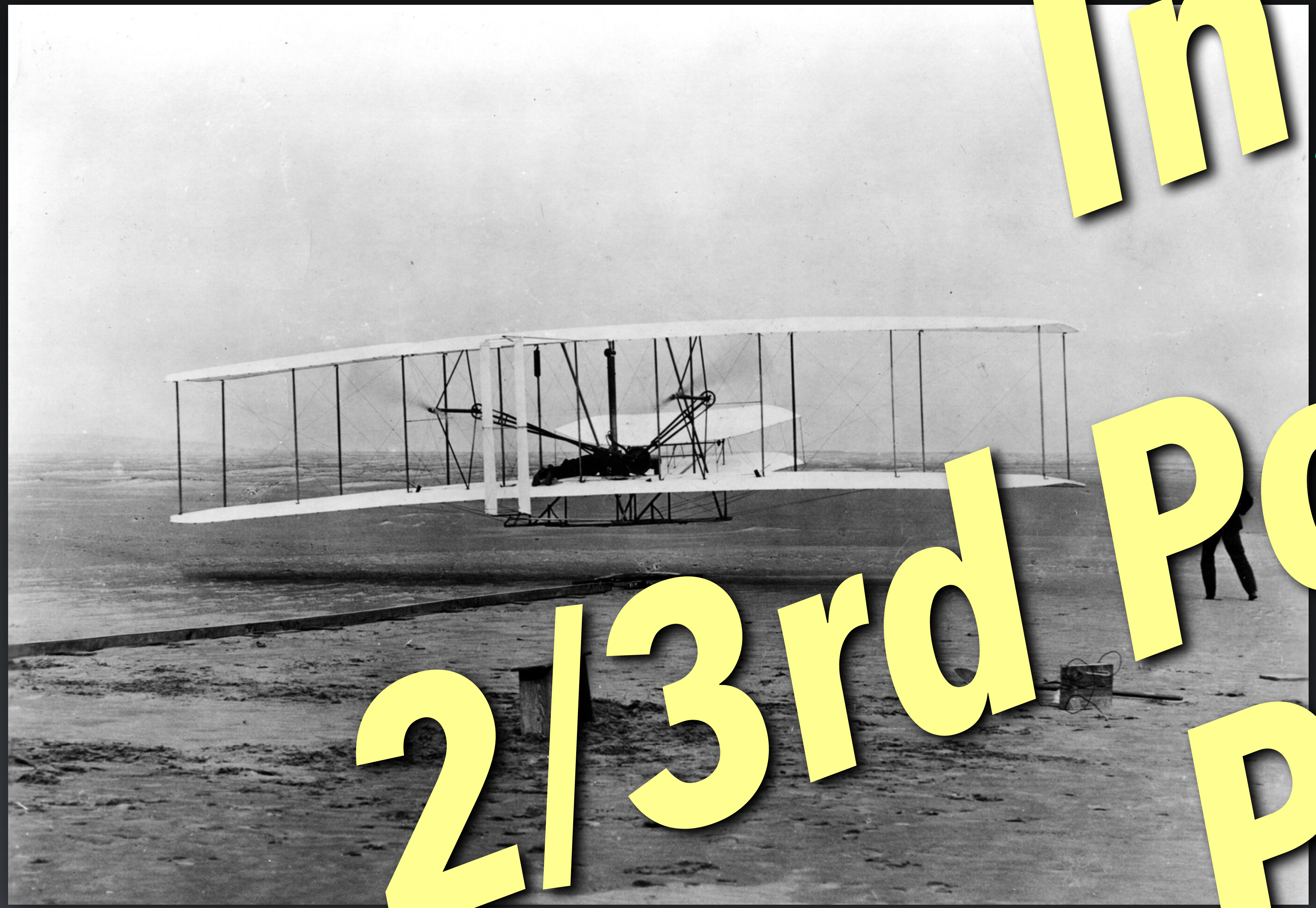
Iteration



Iteration



Iteration



In 2017
2/3rd Population of the
Planet Flew
With no Fatalities!

Continuous Delivery ltd

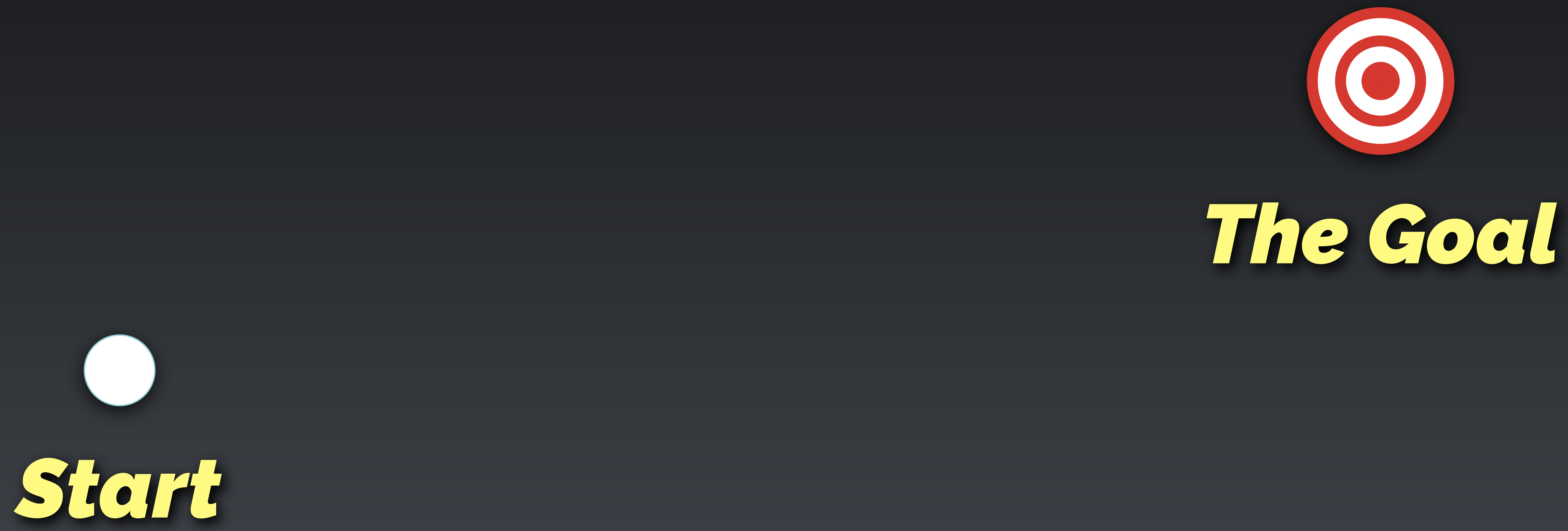
Iteration

Iteration

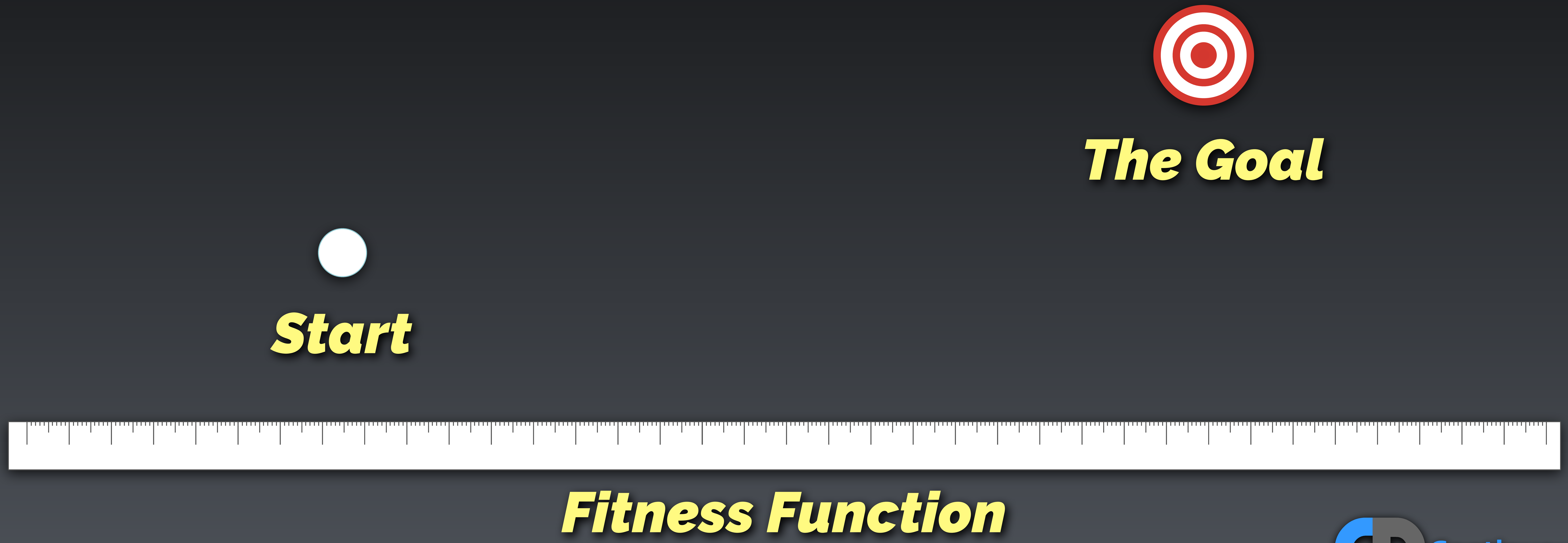
The Goal

Continuous Delivery ltd

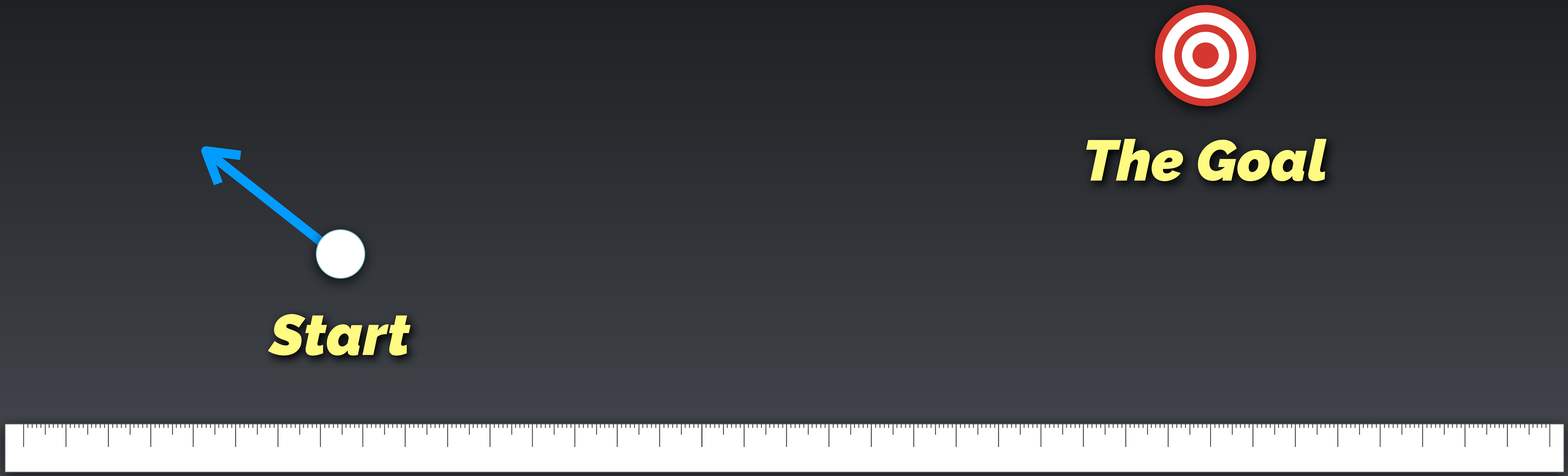
Iteration



Iteration



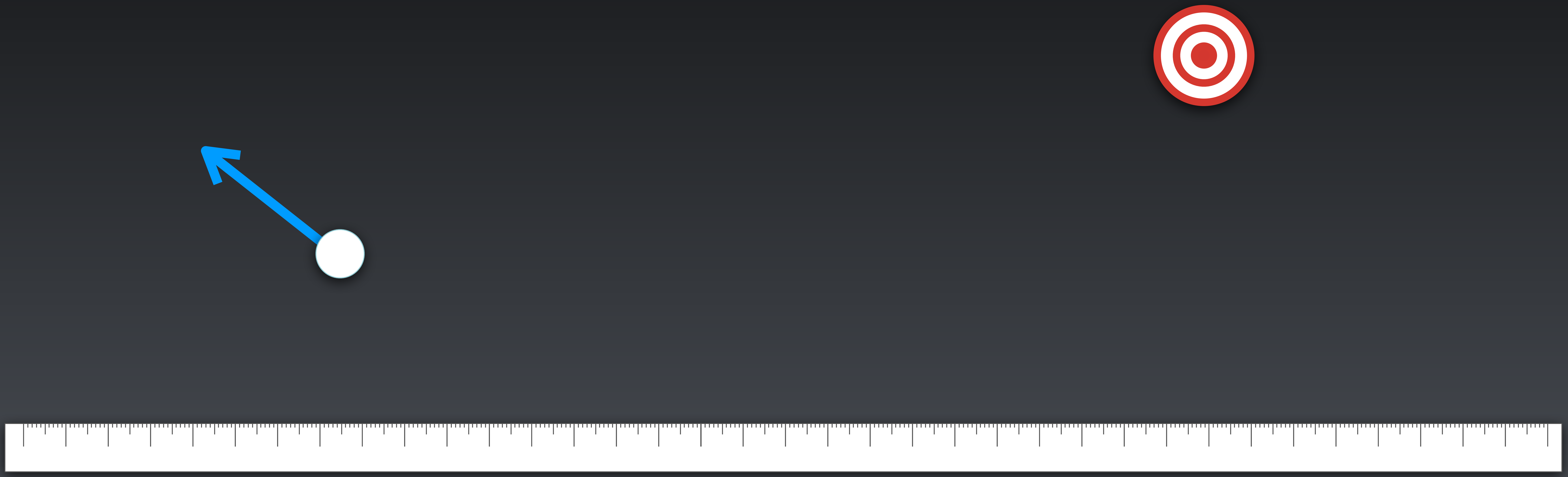
Iteration



Fitness Function

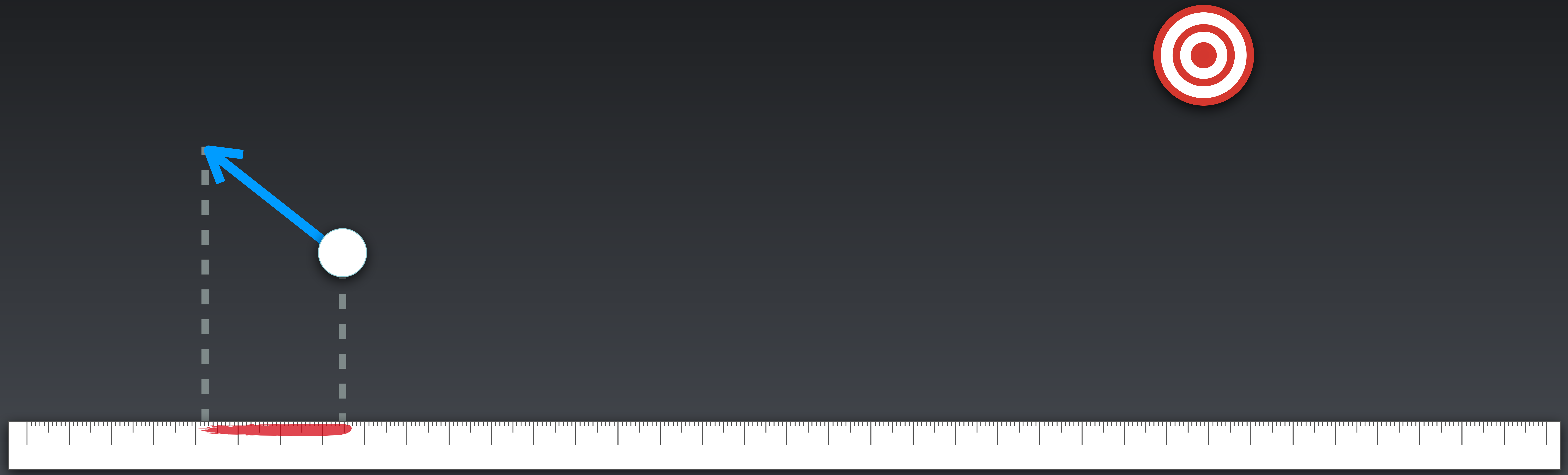
Continuous Delivery ltd

Iteration



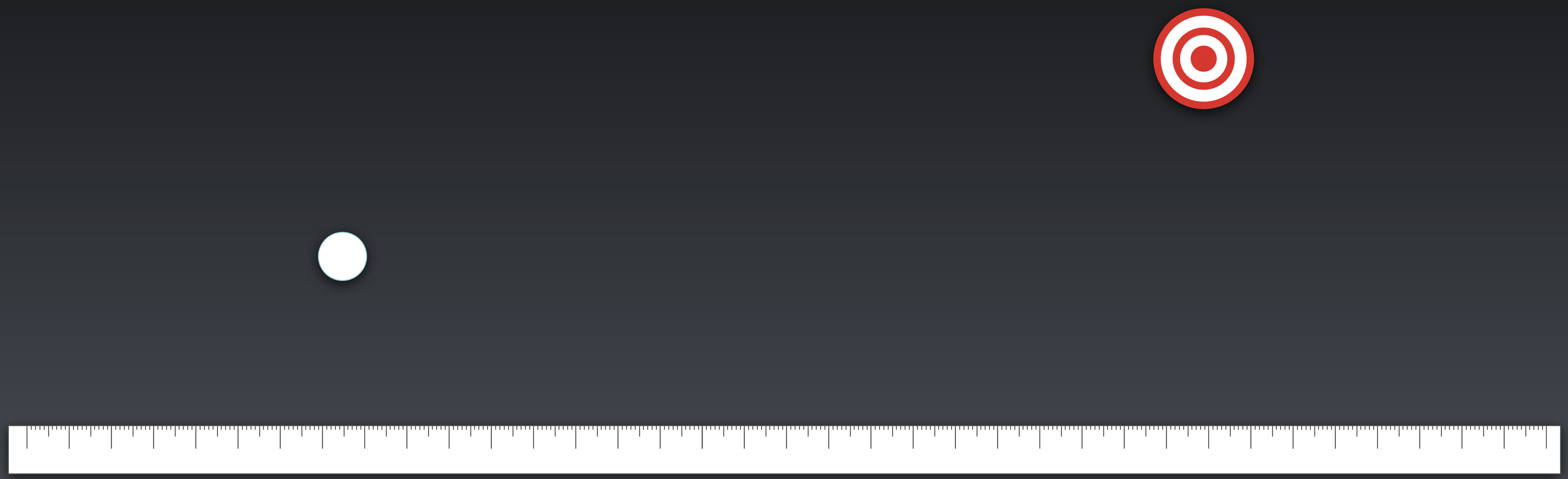
Continuous Delivery ltd

Iteration

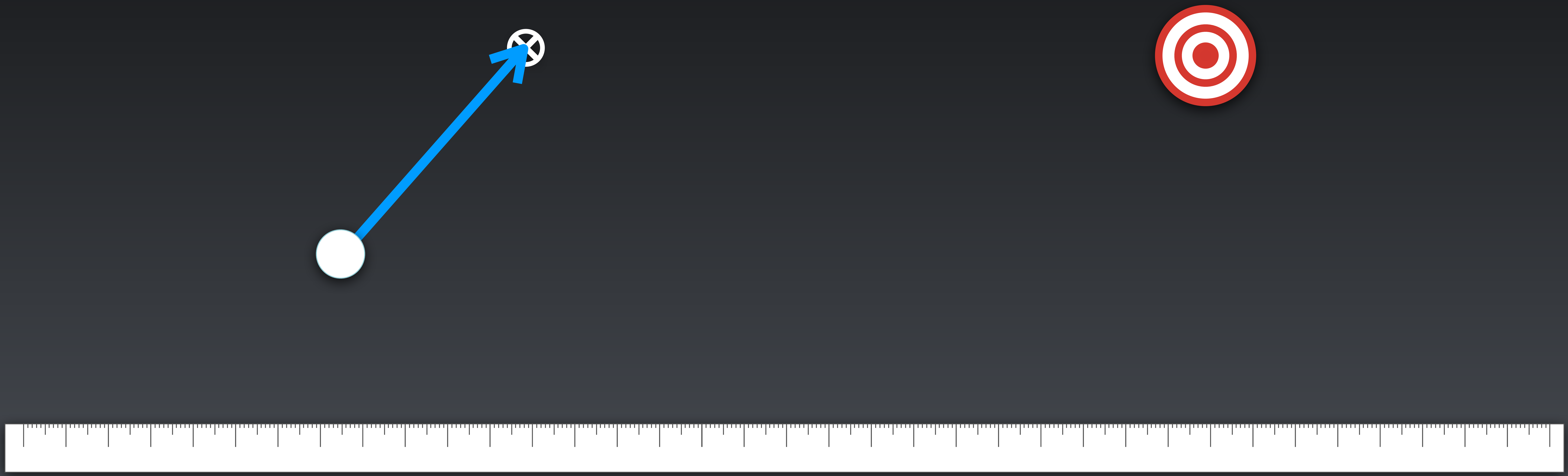


Continuous Delivery ltd

Iteration

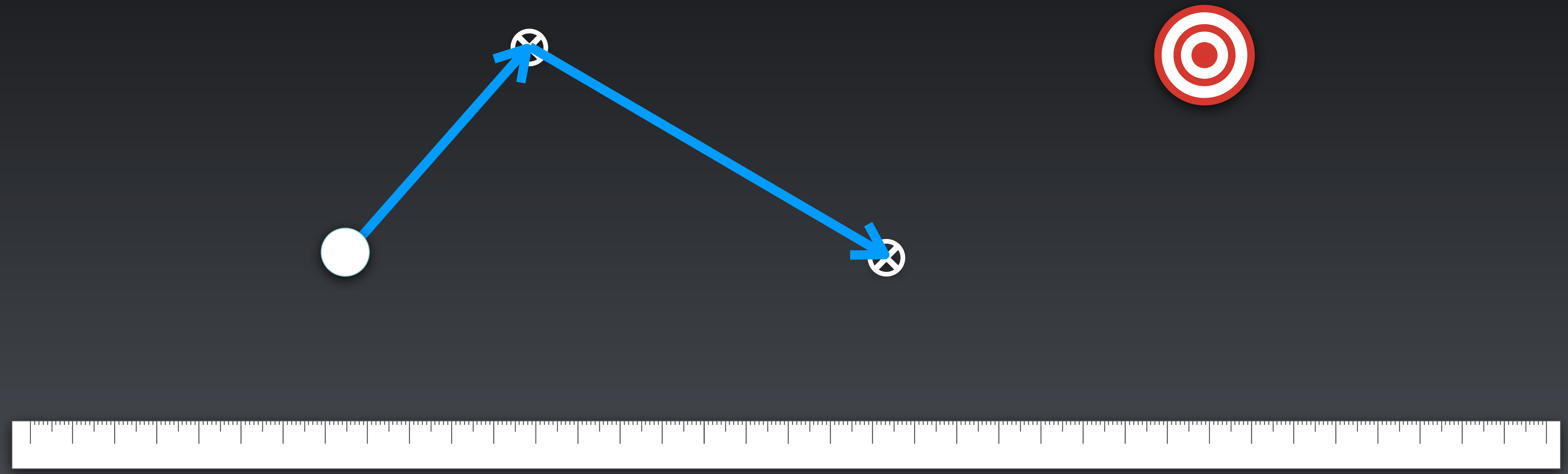


Iteration



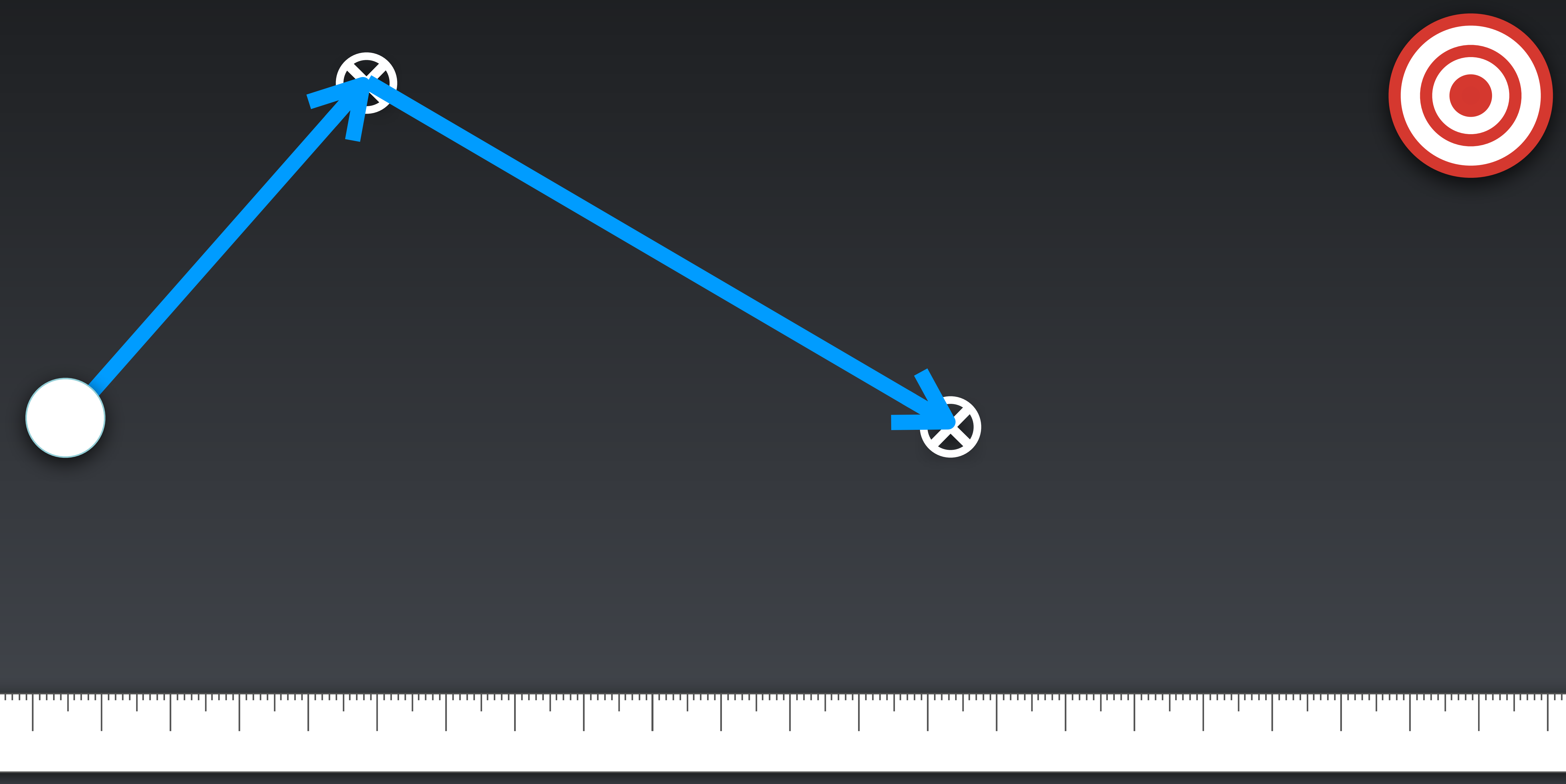
Continuous Delivery ltd

Iteration



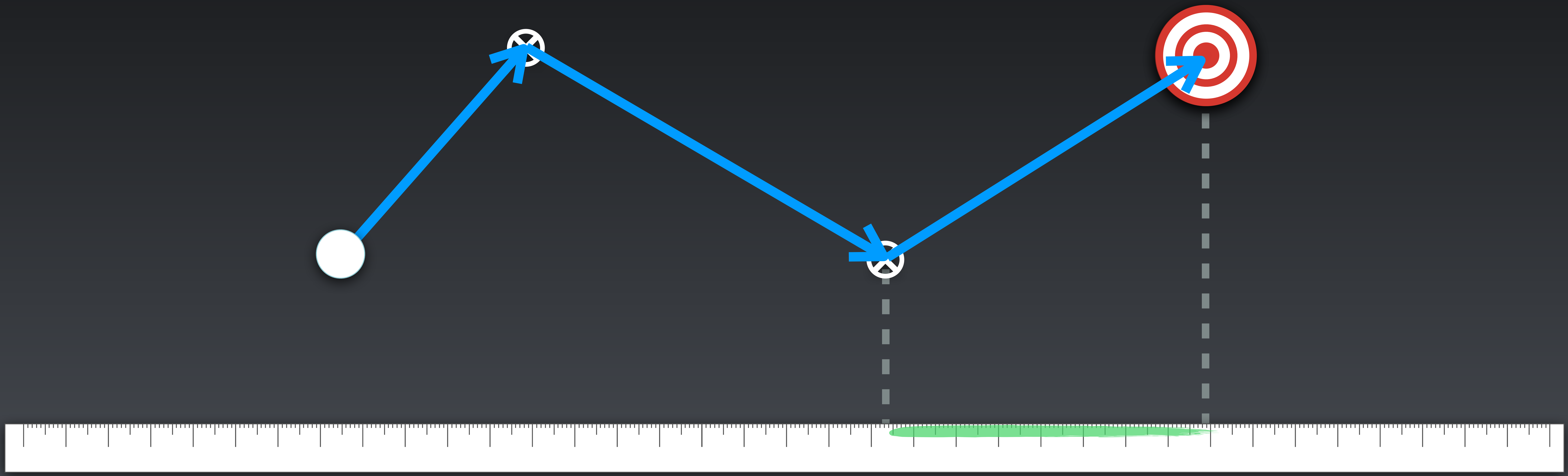
Continuous Delivery ltd

Iteration



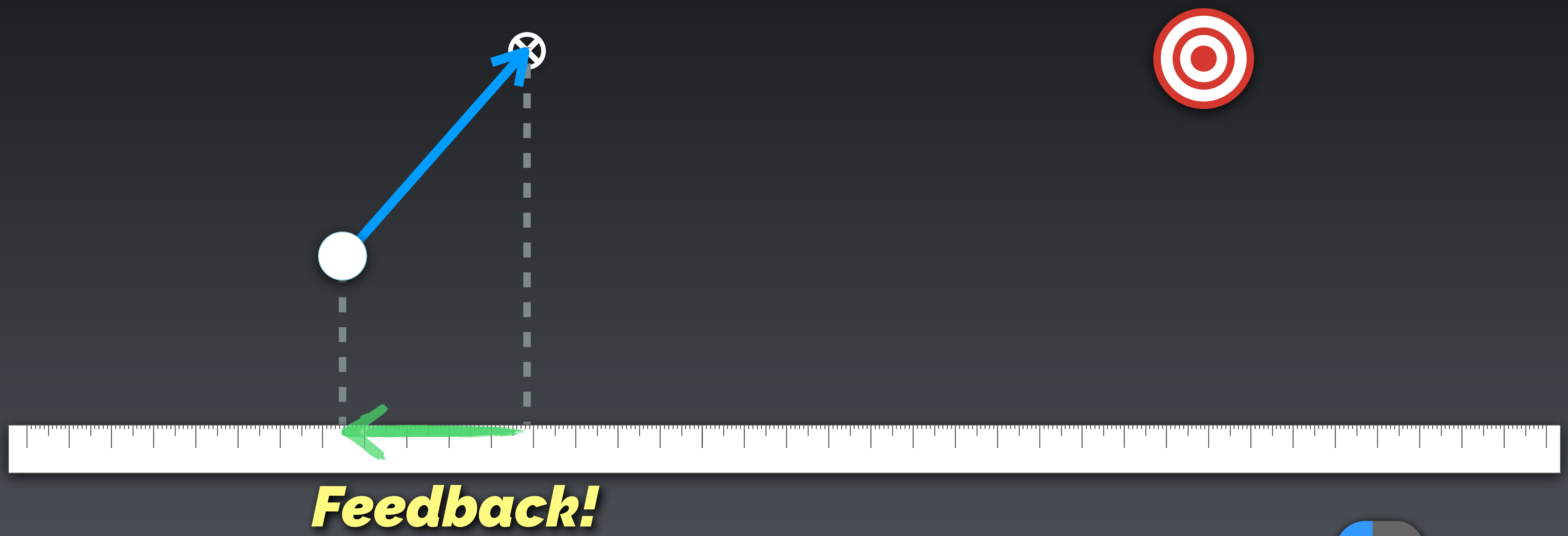
Continuous Delivery ltd

Iteration



Continuous Delivery ltd

Feedback



Continuous Delivery ltd

Feedback

Feedback

How Do You Balance a Broom?

Continuous Delivery ltd

Feedback

Continuous Delivery ltd

Feedback

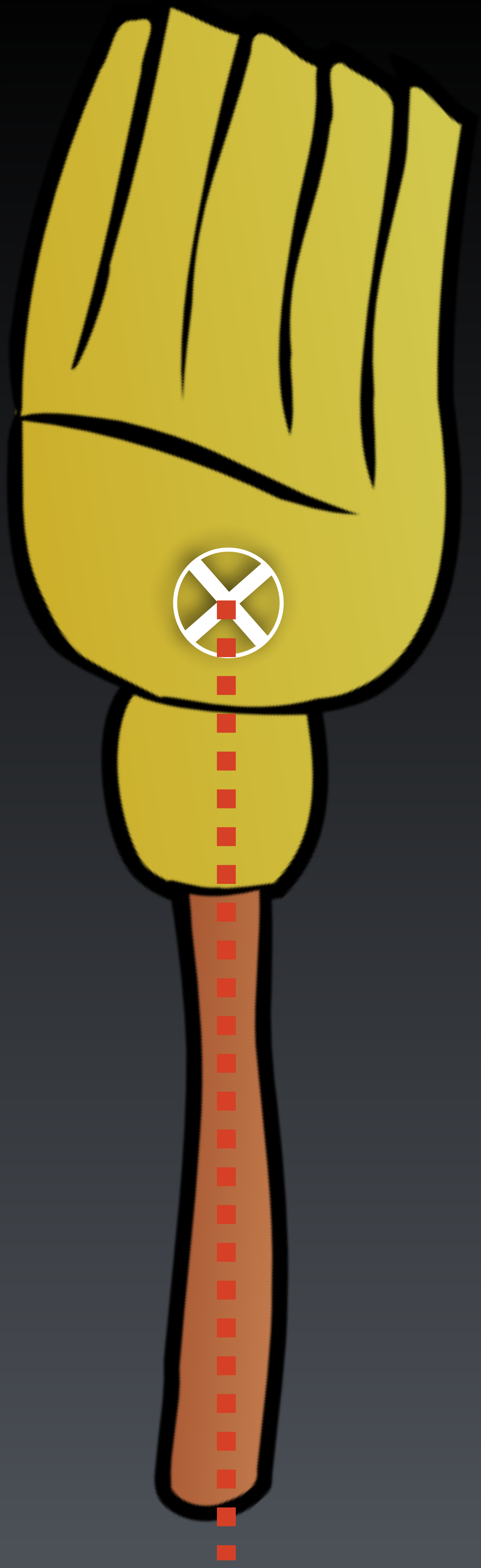
Continuous Delivery ltd

Feedback

Continuous Delivery ltd

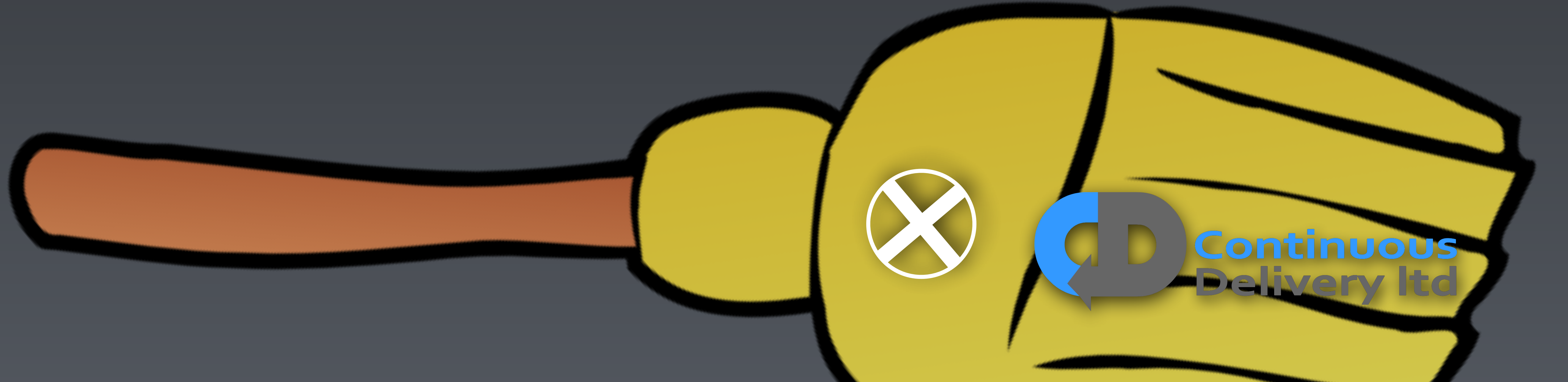
Feedback

Feedback



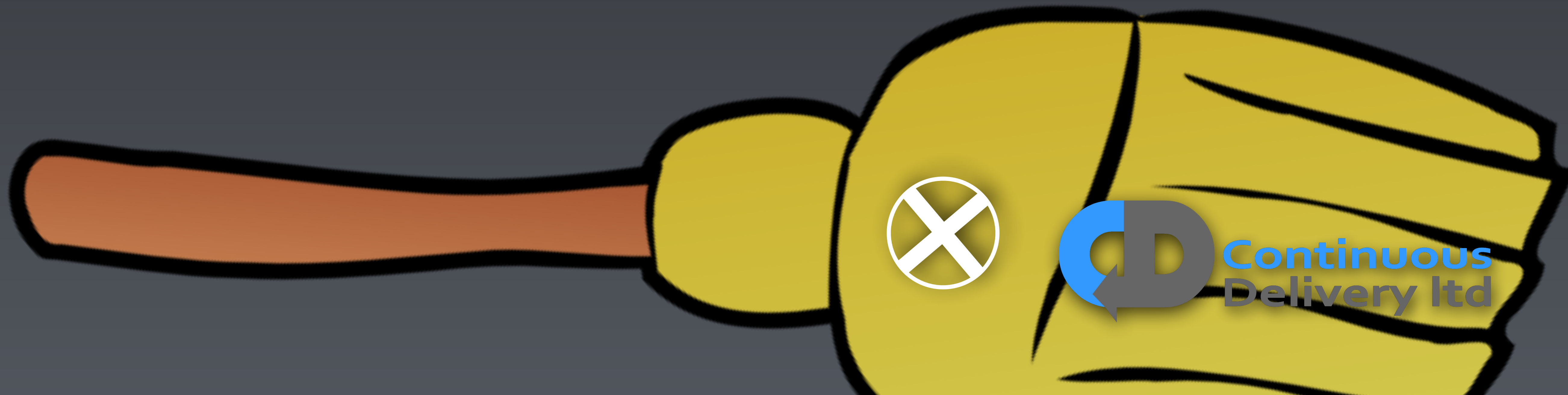
Feedback

There's Only One Solution to This!



Continuous Delivery ltd

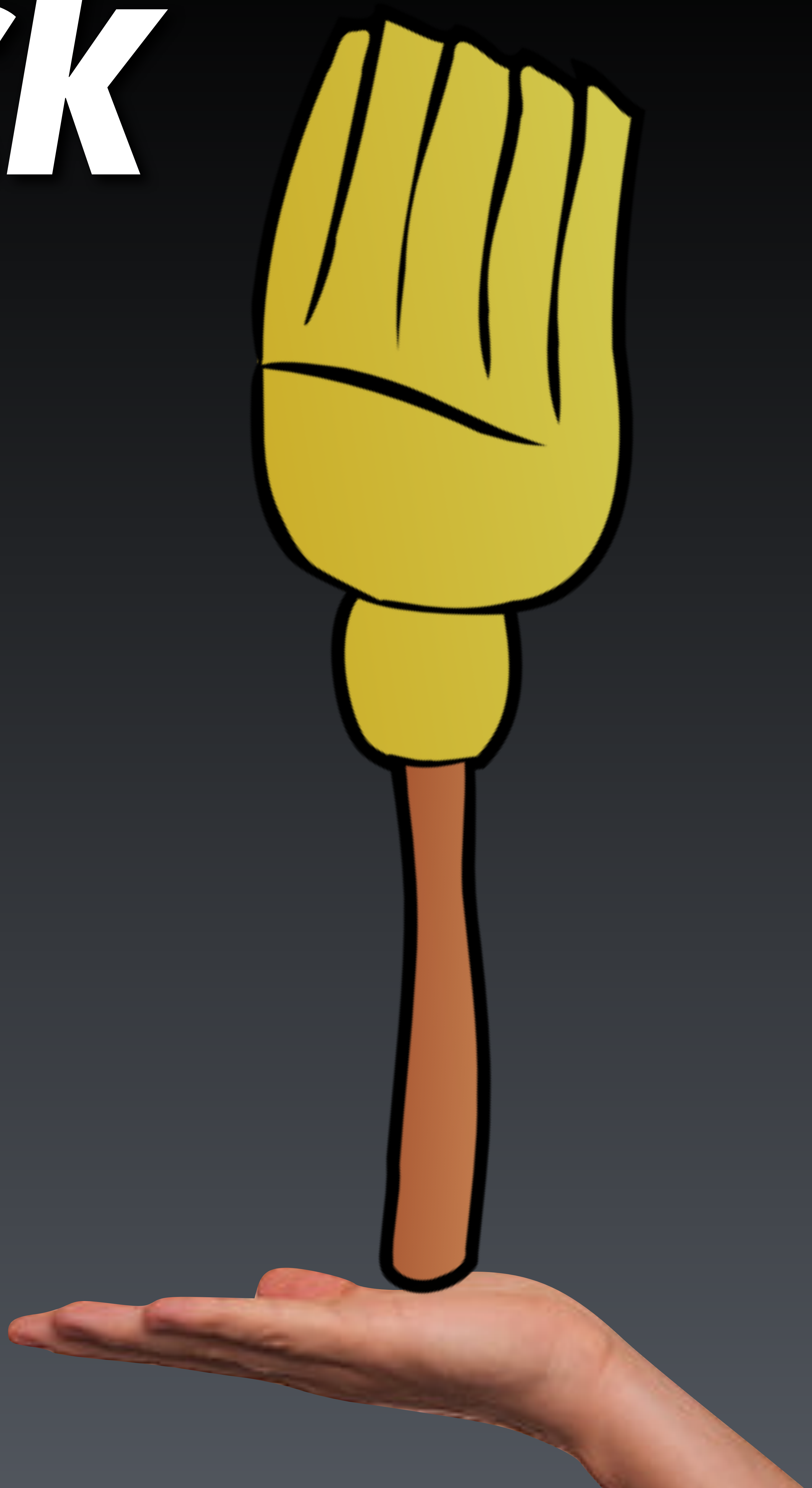
Feedback



Continuous Delivery ltd

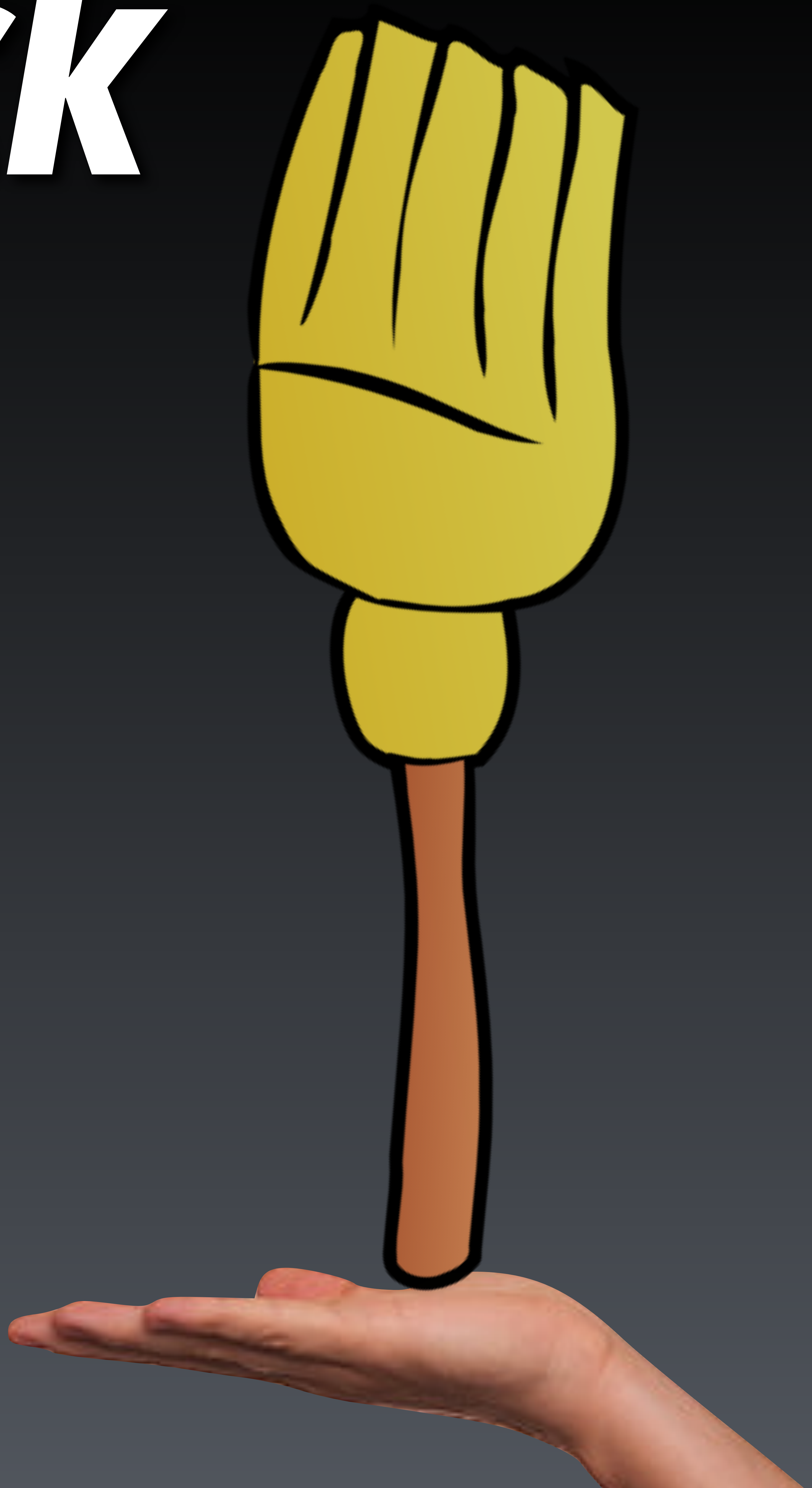
Feedback

Feedback

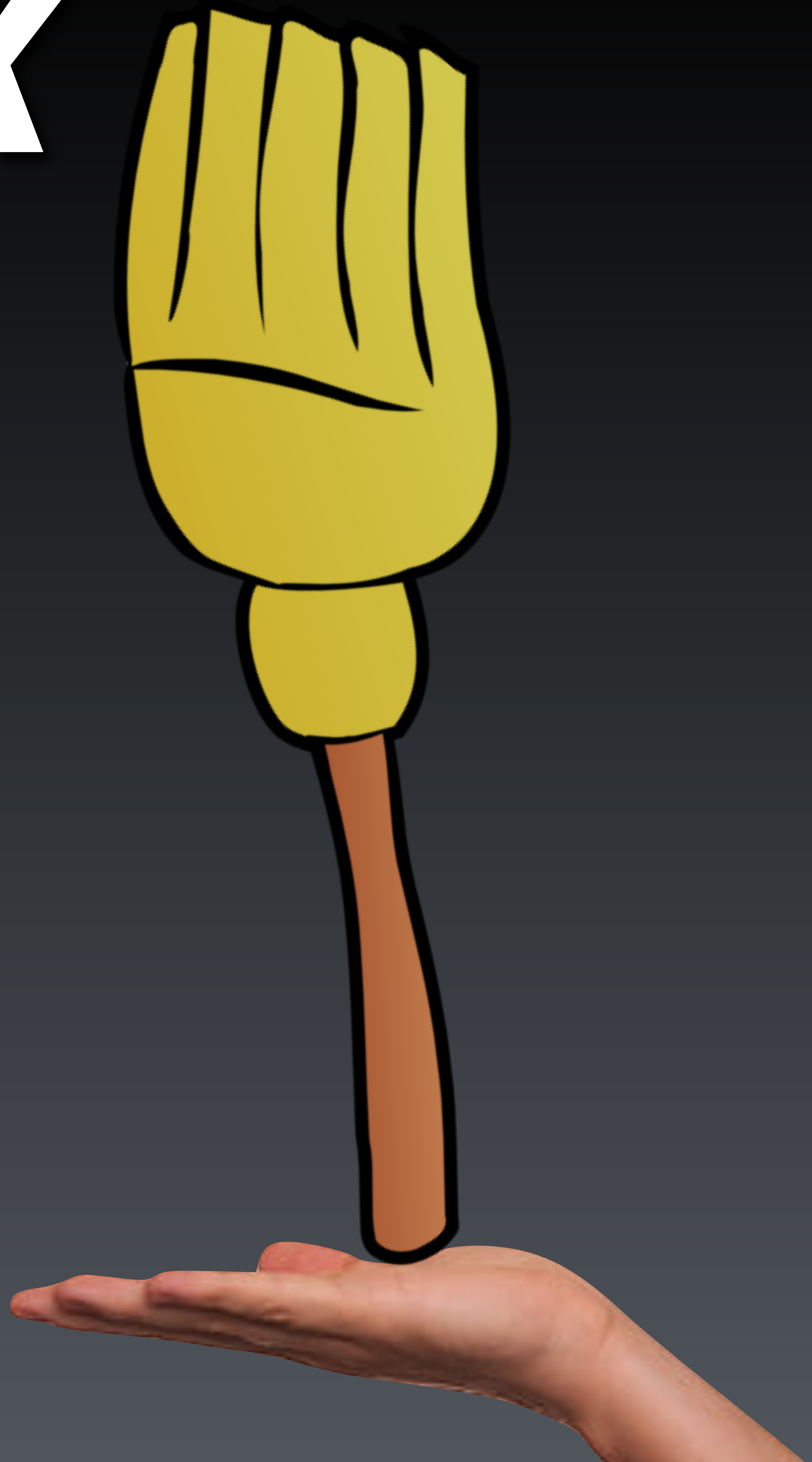


Continuous Delivery ltd

Feedback

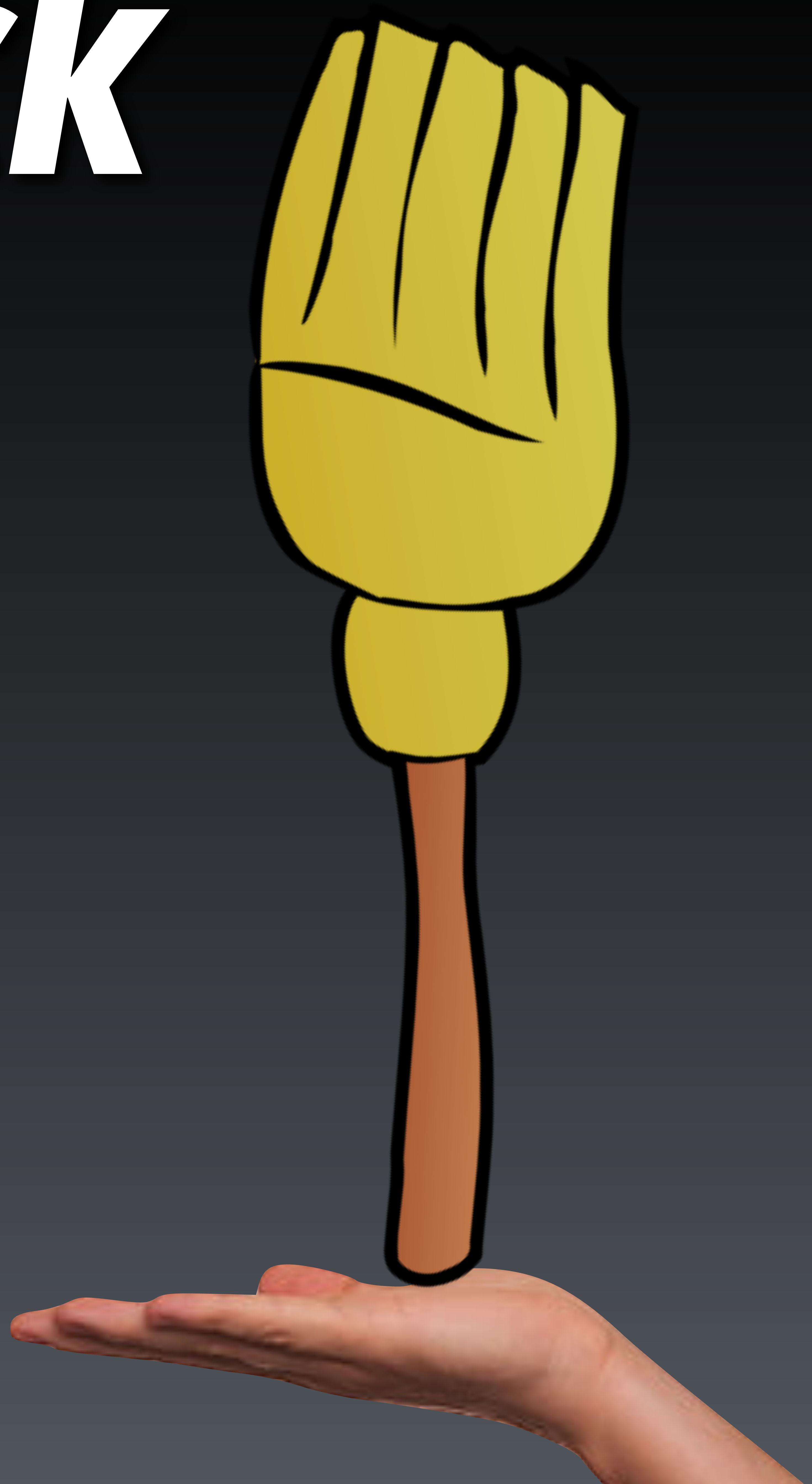


Feedback



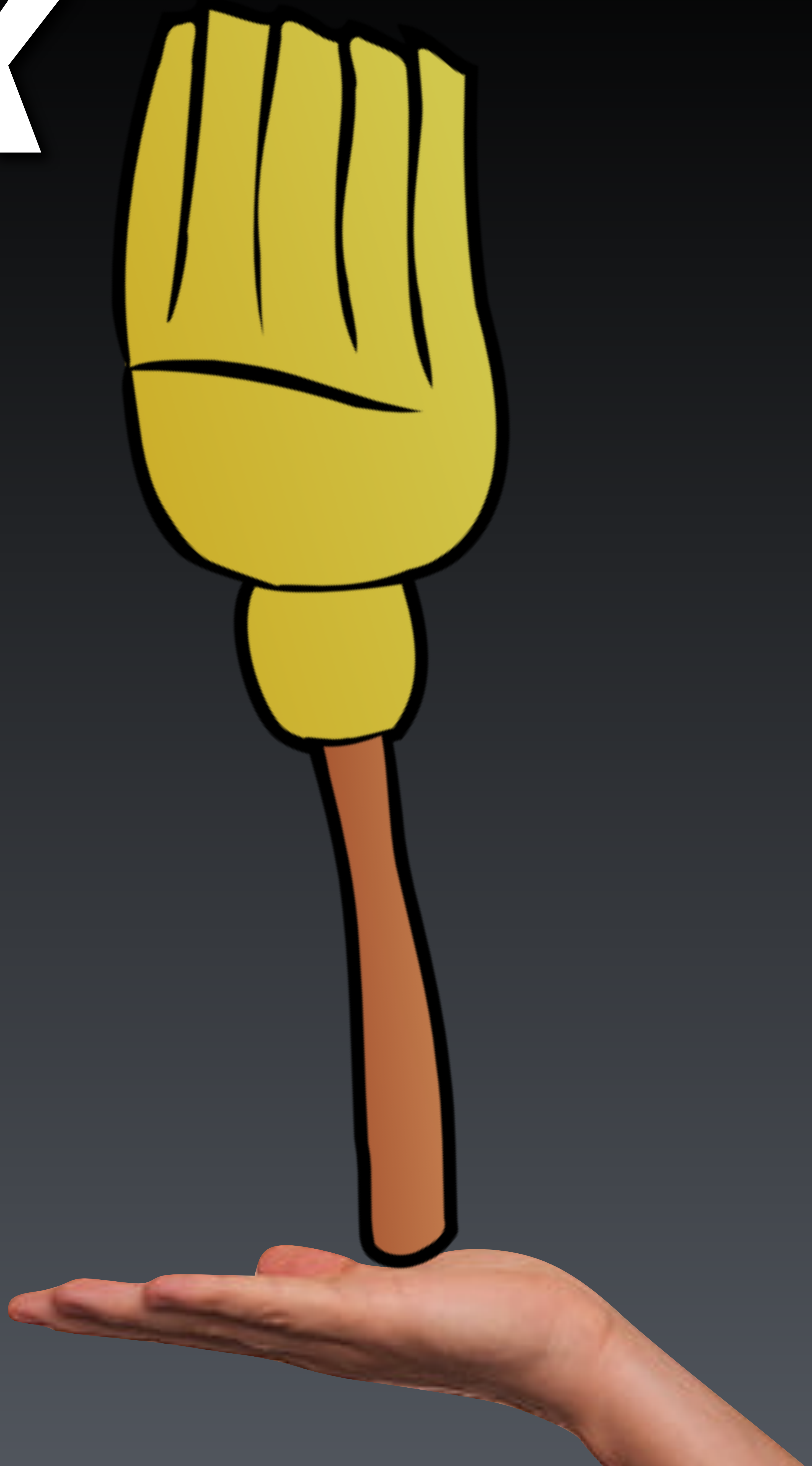
Continuous Delivery ltd

Feedback



Continuous Delivery ltd

Feedback



Continuous Delivery ltd

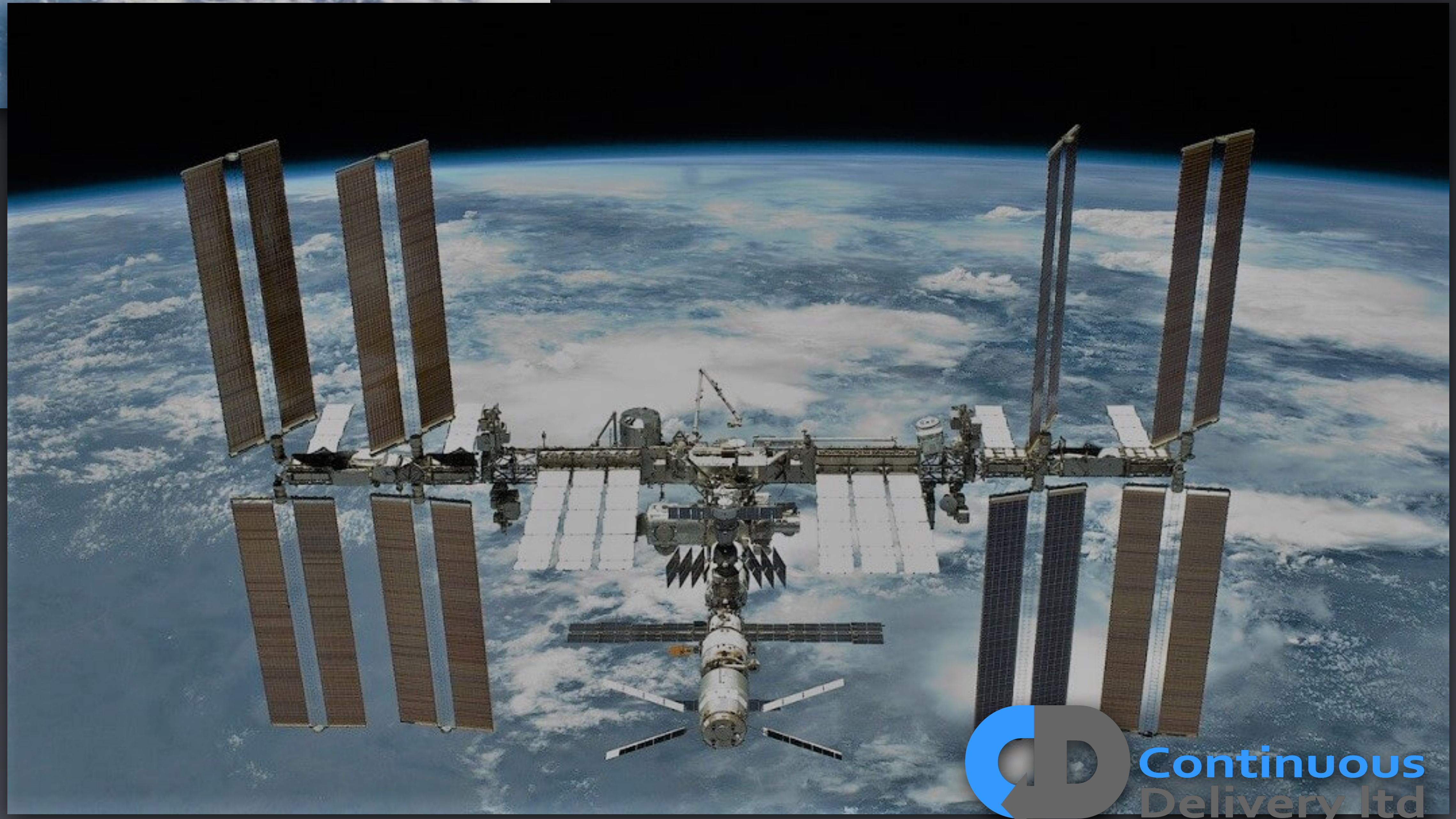
Feedback

Continuous Delivery ltd

Incremental

Incremental

Incremental



Continuous Delivery ltd

Experimental

Scary!

Experimental

How Else Can We Make Progress?

Experimental

Experimental

Experimental

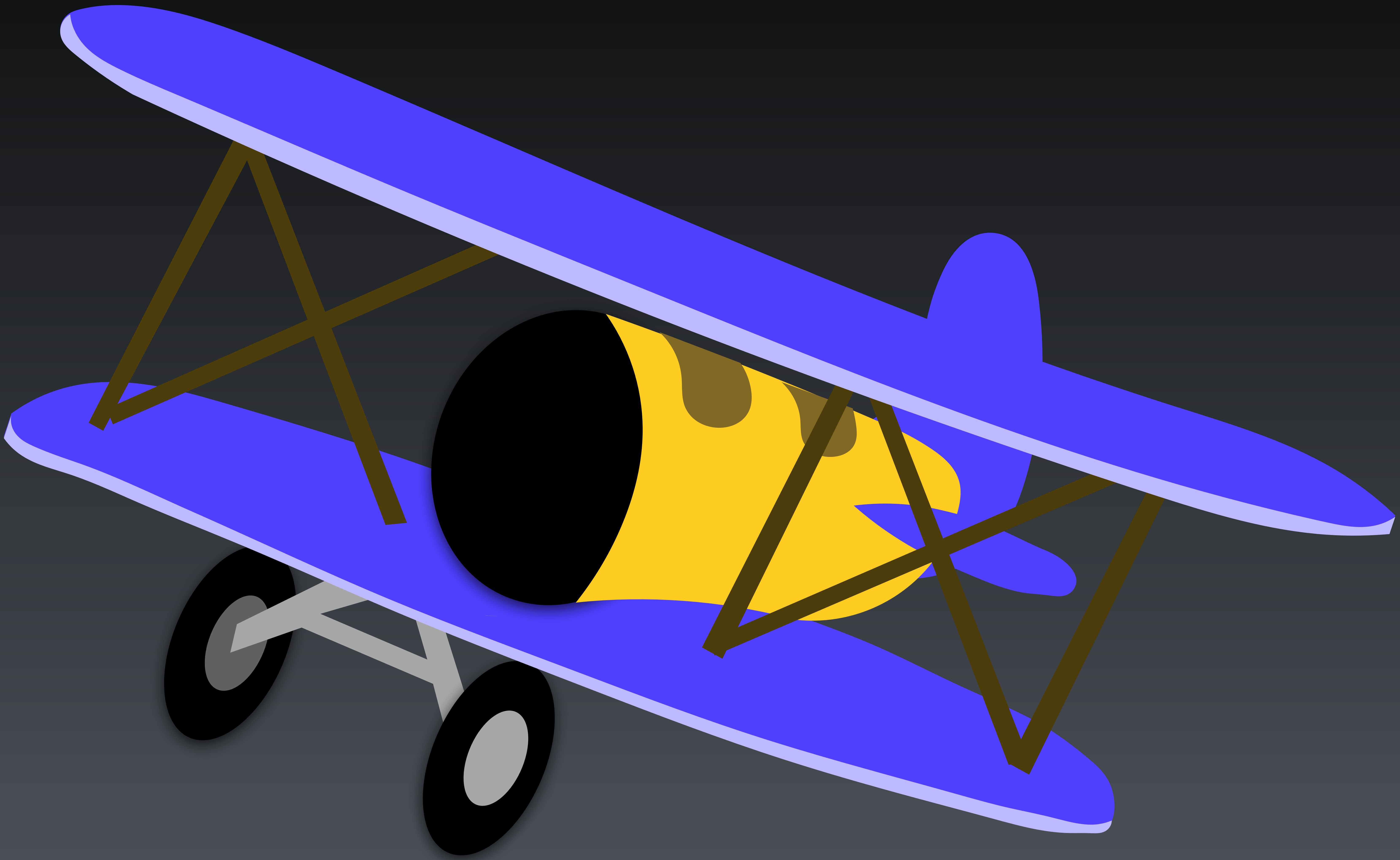
Controlling the Variables!

Continuous Delivery ltd

Experimental

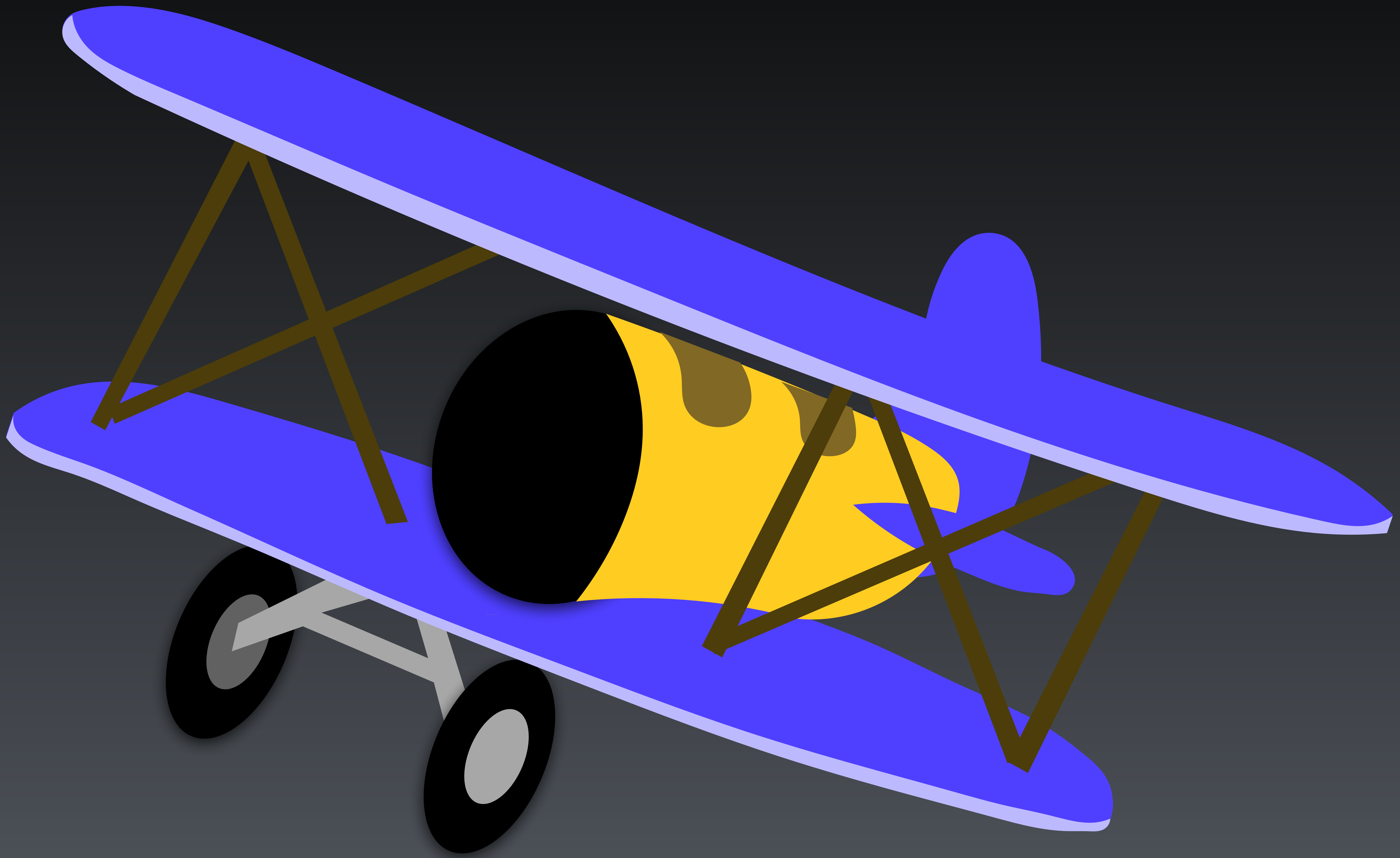
Continuous Delivery ltd

Experimental



Continuous Delivery ltd

Experimental



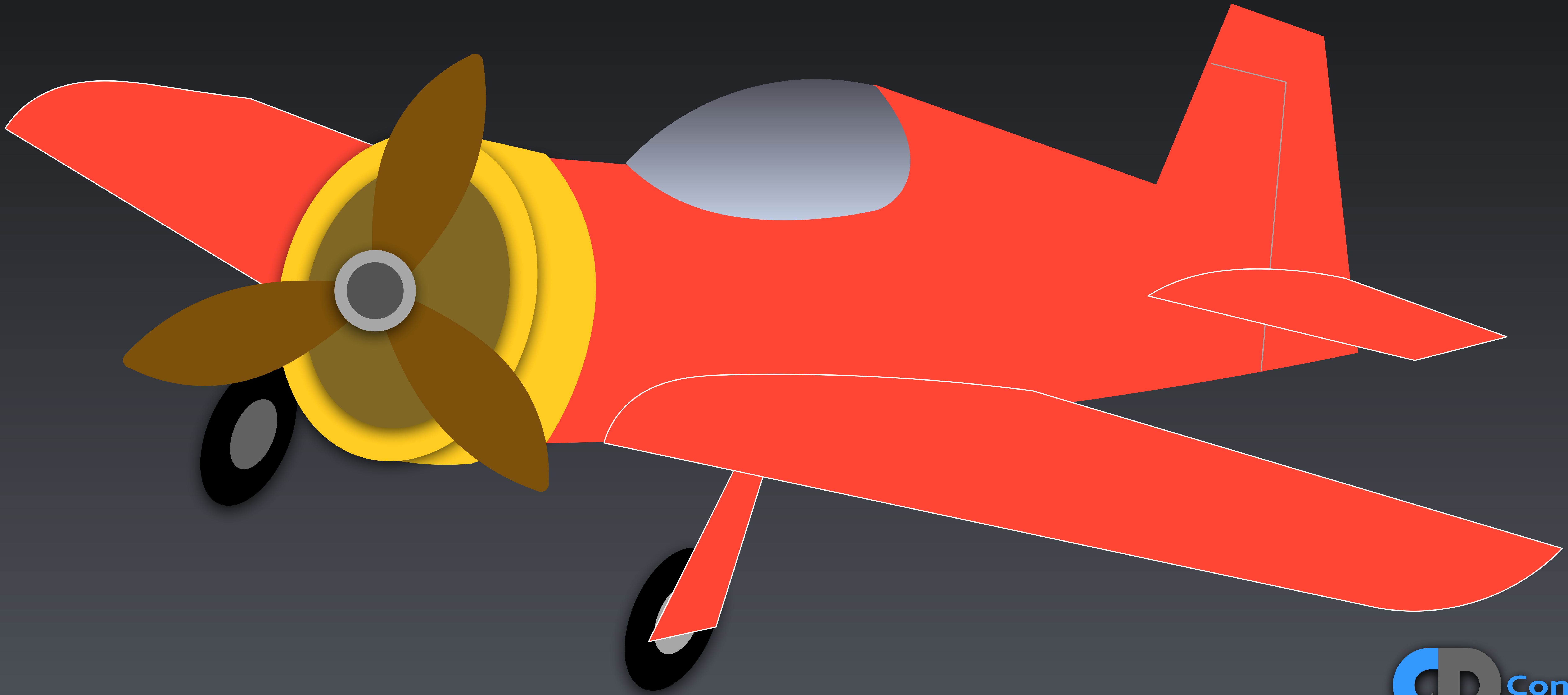
Continuous Delivery ltd

Experimental

Continuous Delivery ltd

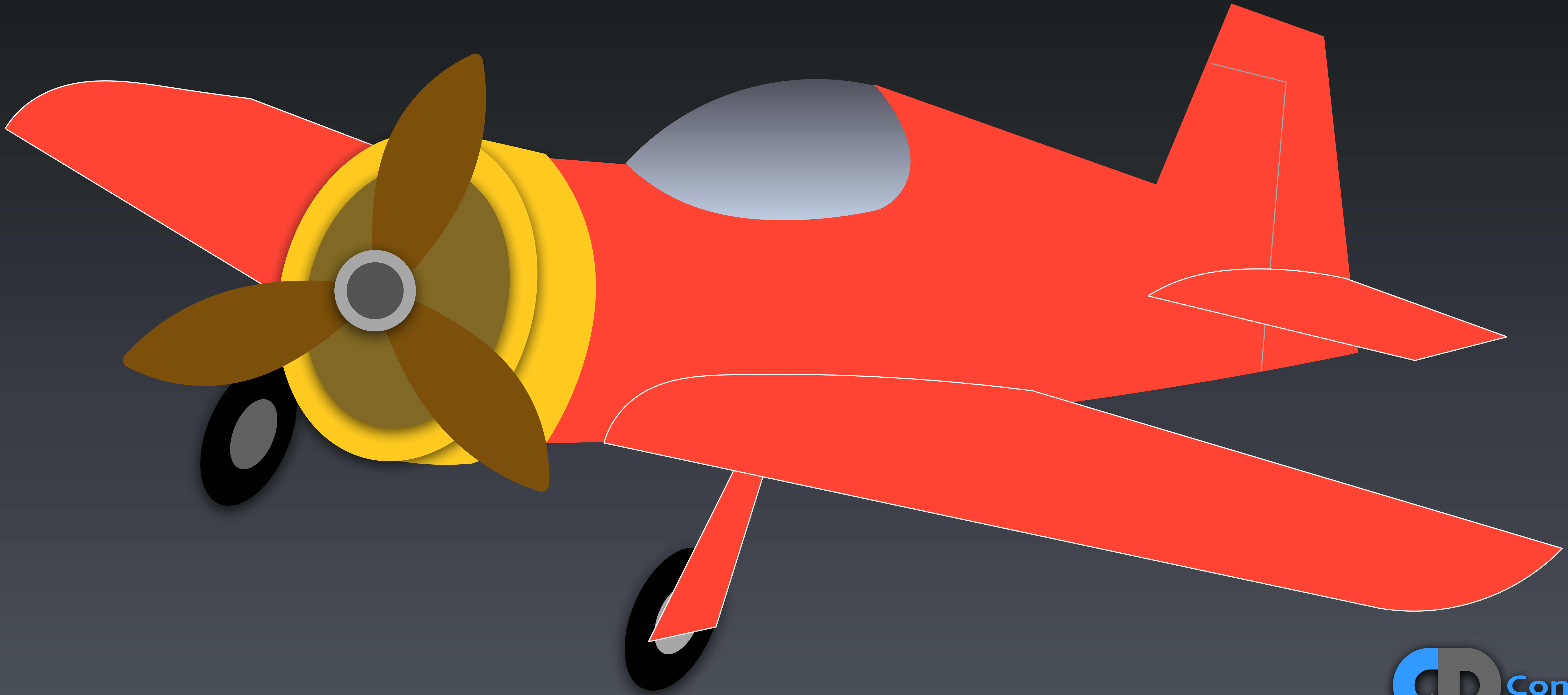
Experimental

Experimental



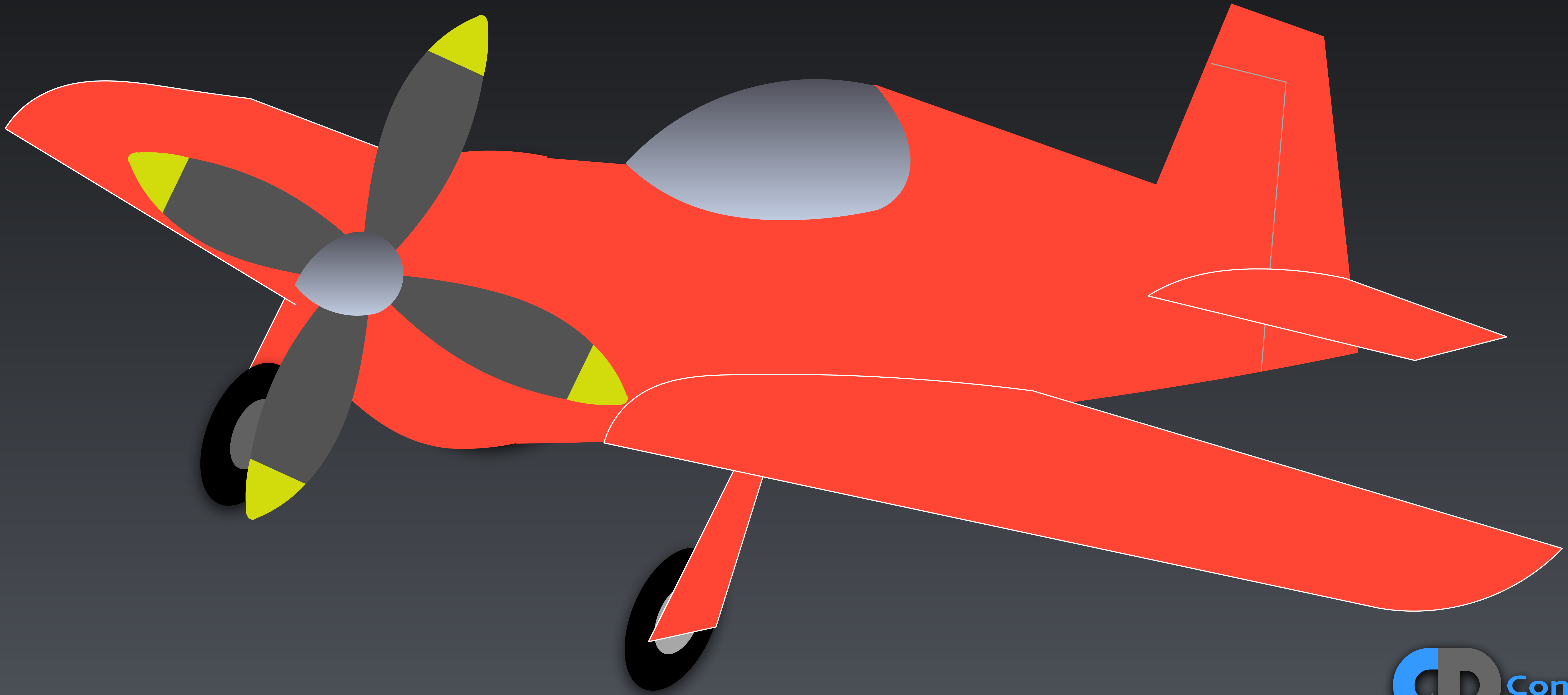
Continuous Delivery ltd

Experimental



Continuous Delivery ltd

Experimental



Empirical

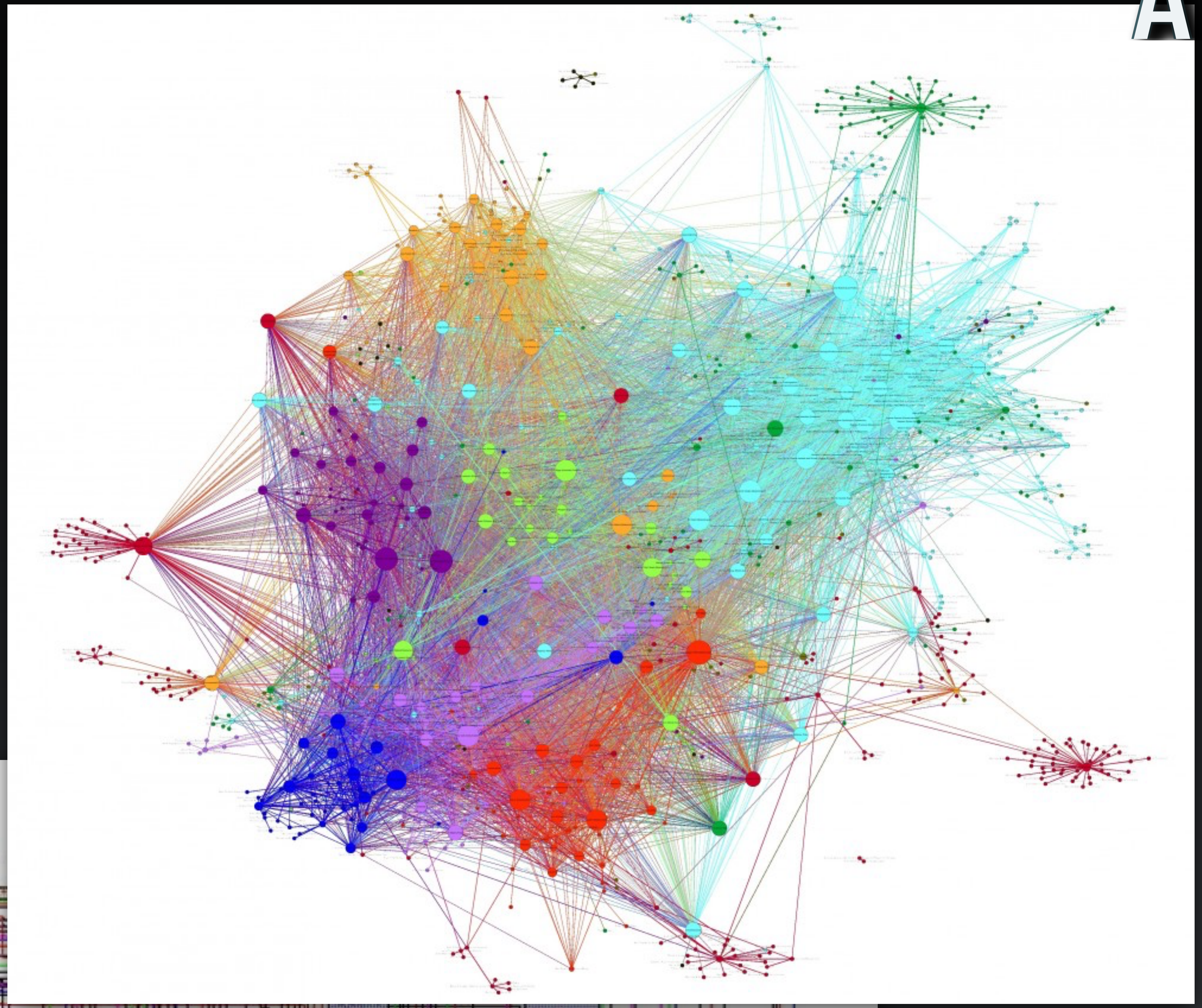
Continuous Delivery ltd

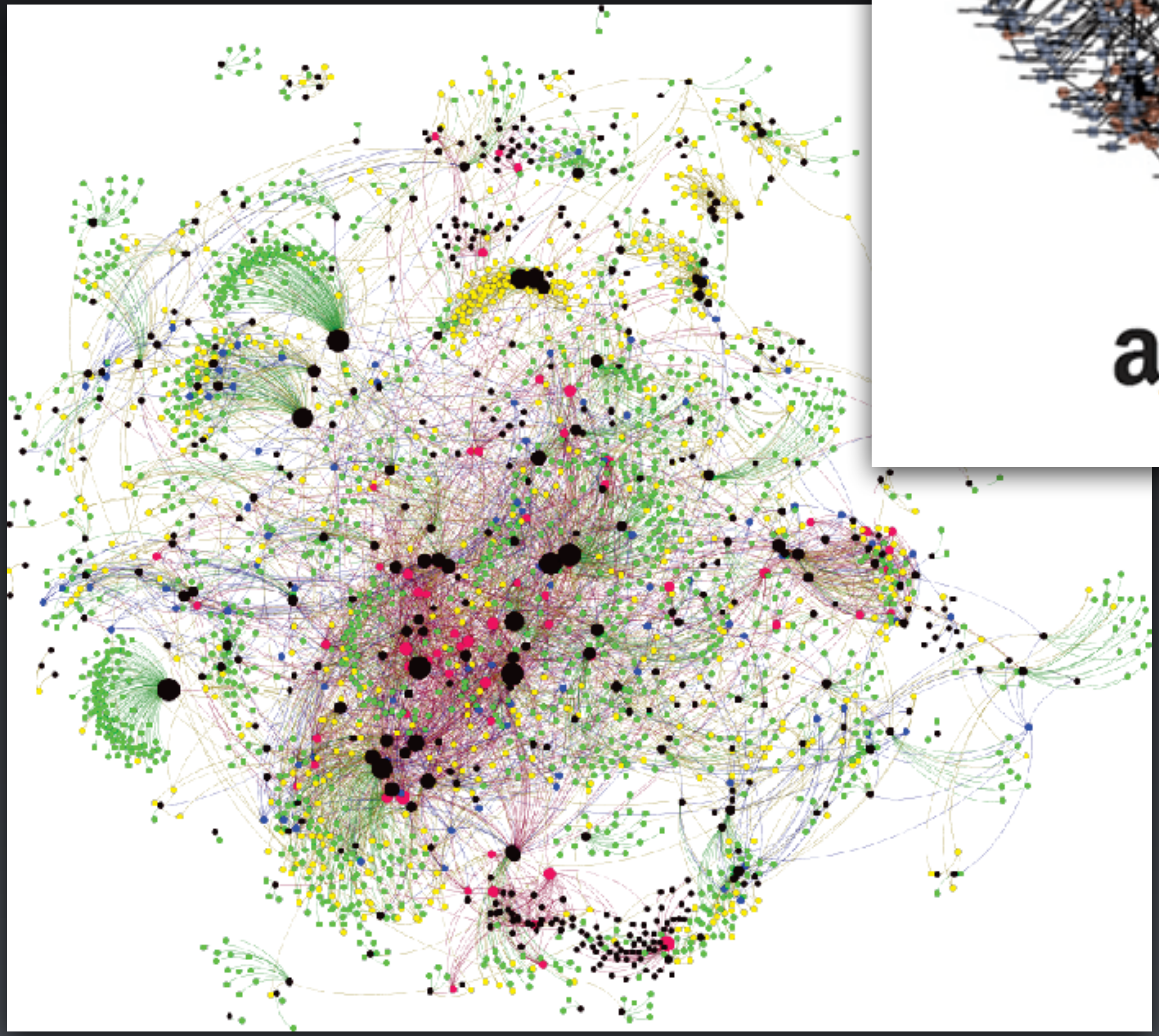
Empirical

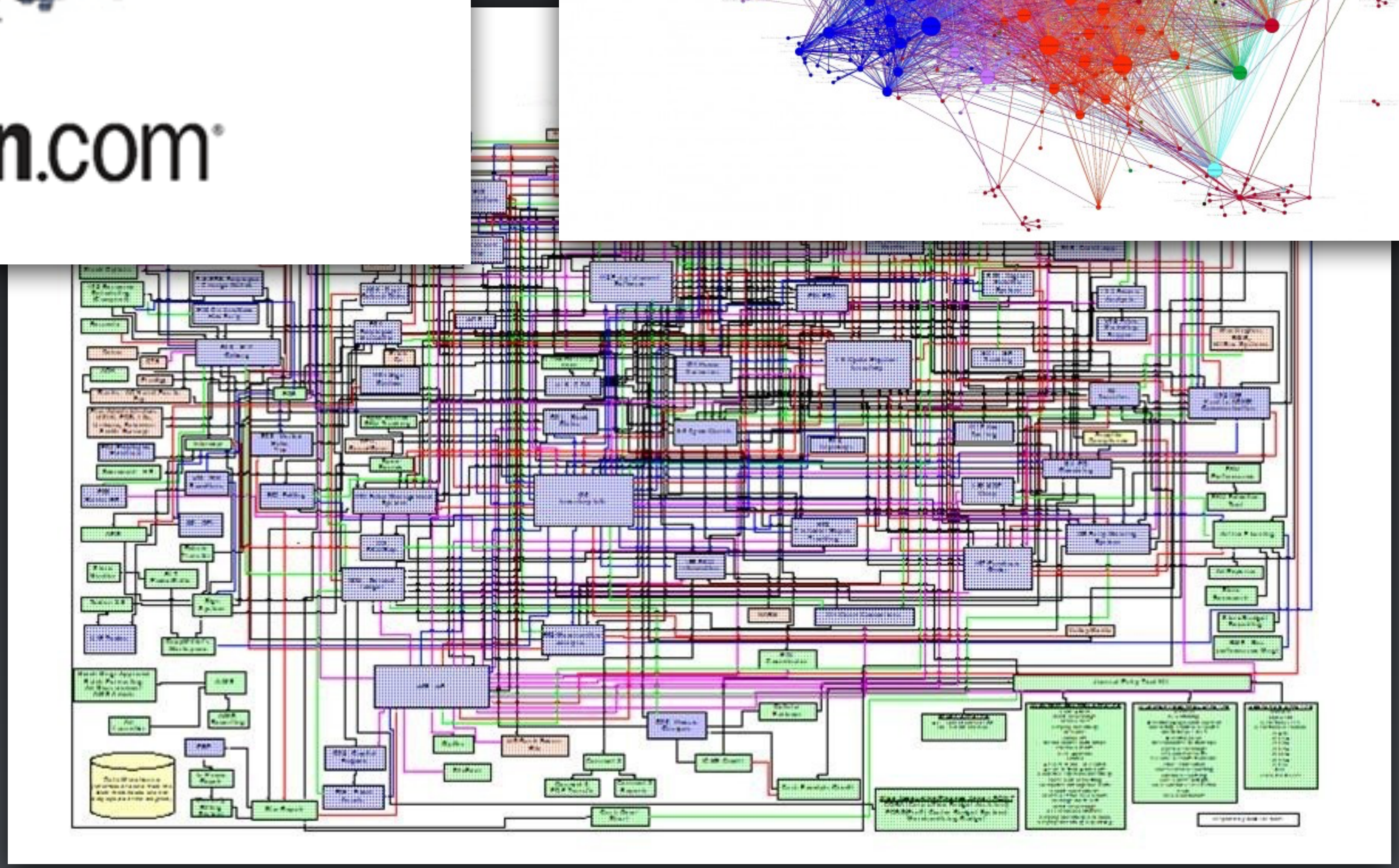
Continuous Delivery ltd

Empirical

Continuous Delivery ltd

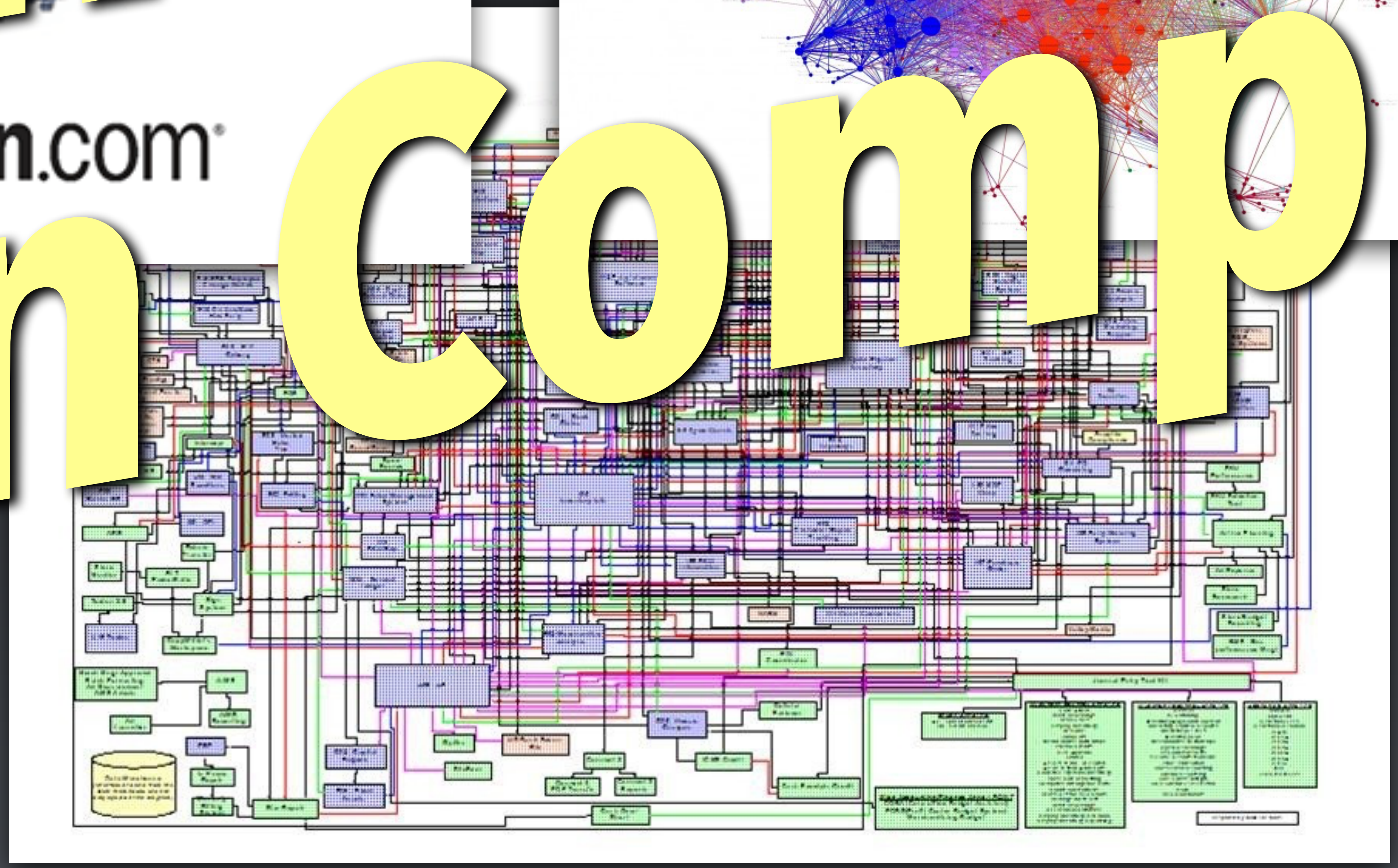






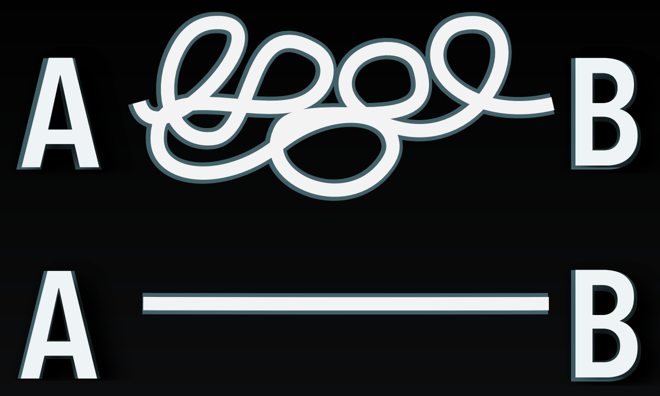
Continuous Delivery ltd

Modern Software is Often Complex!

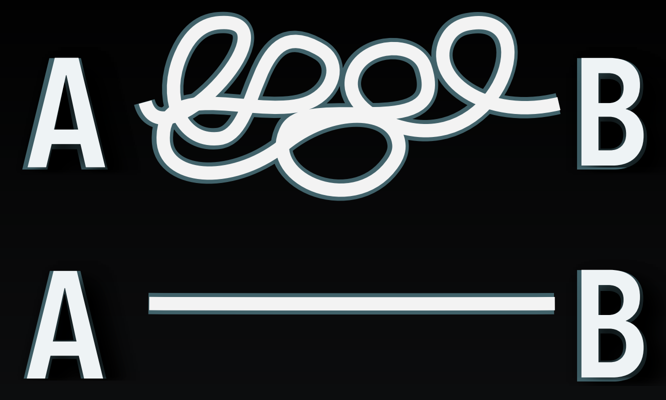


Software Development is about Learning & Managing Complexity

Continuous Delivery ltd



Managing Complexity



Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

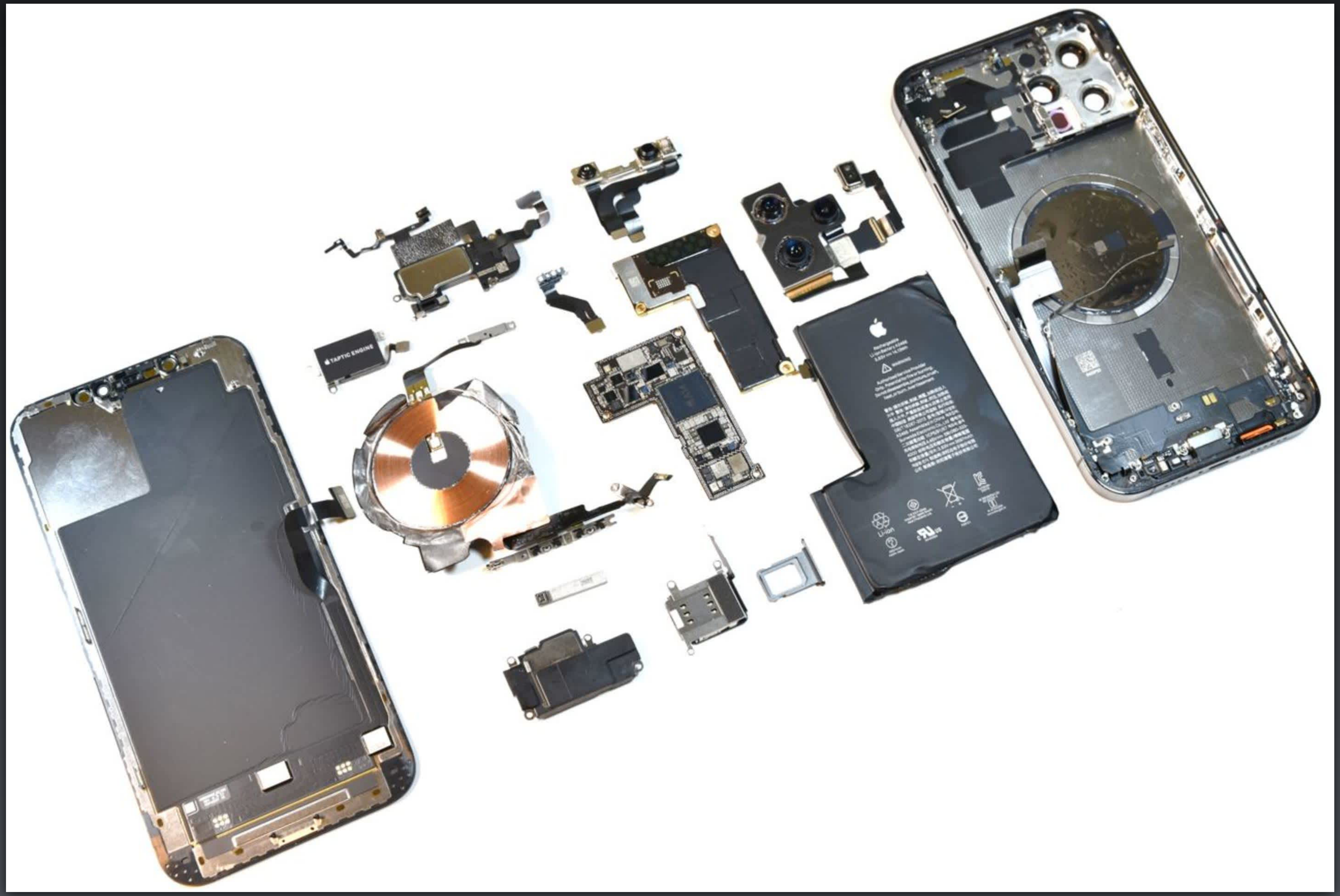
Coupling

Continuous Delivery ltd

Modularity

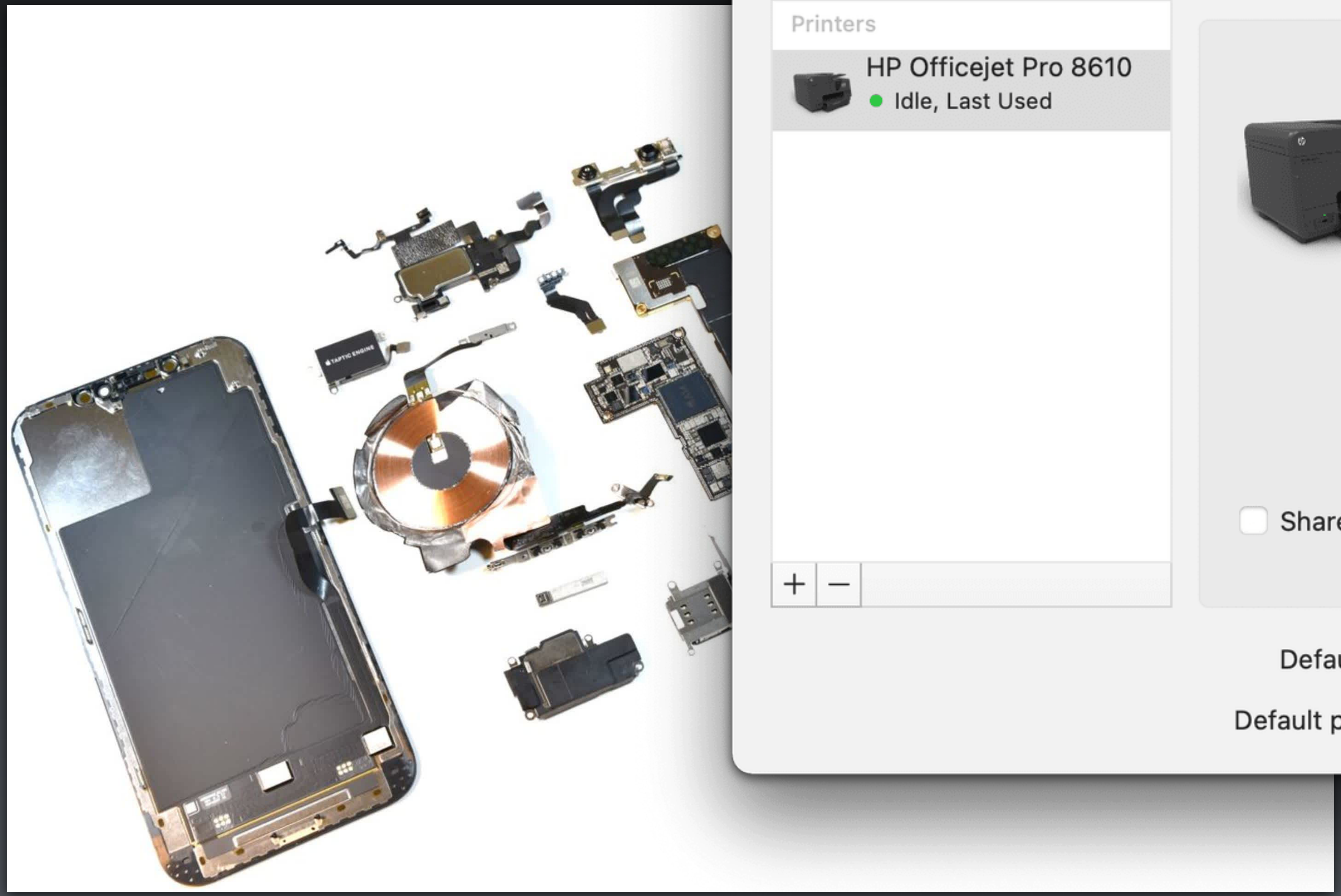
Modularity

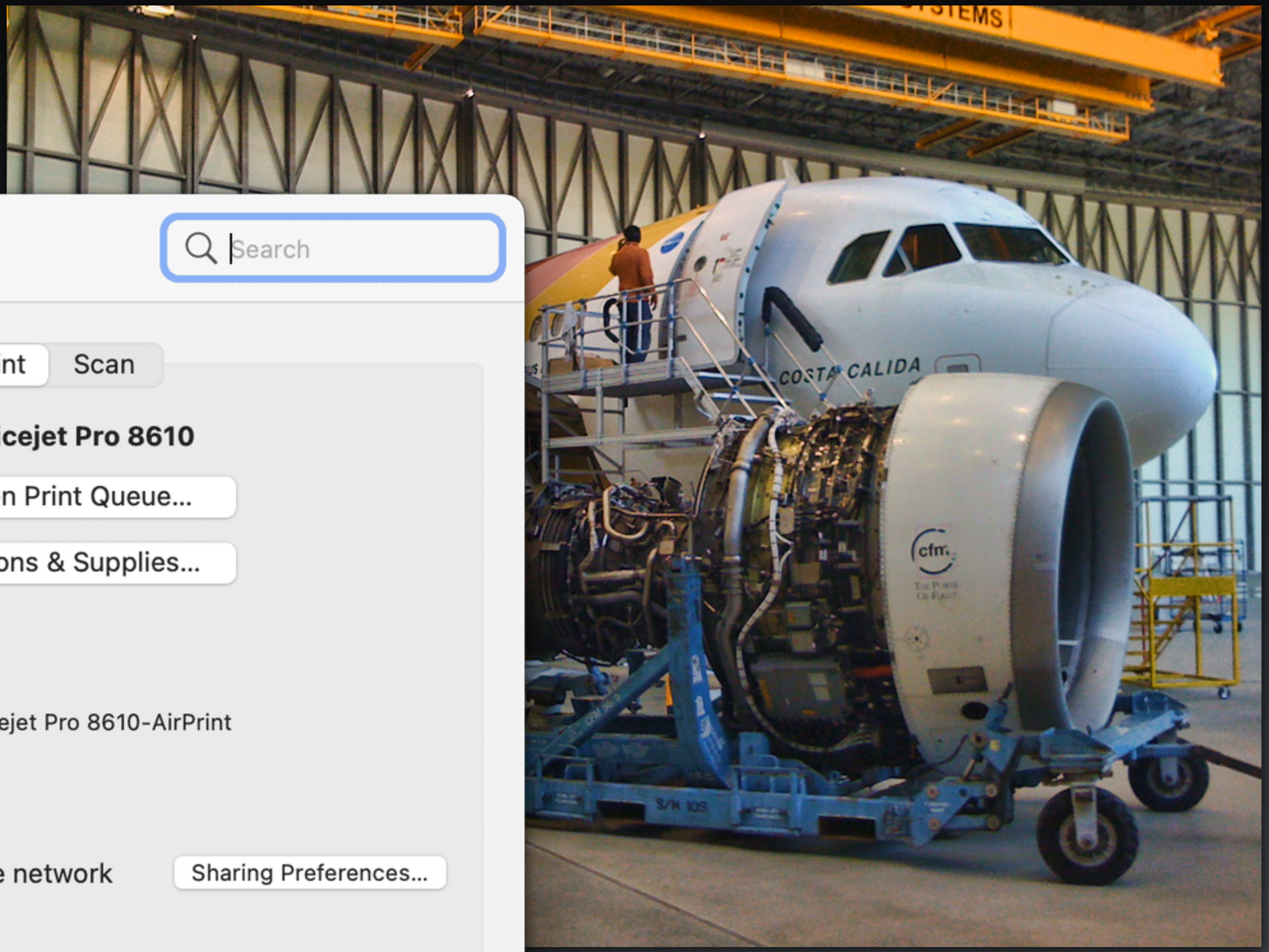
Modularity

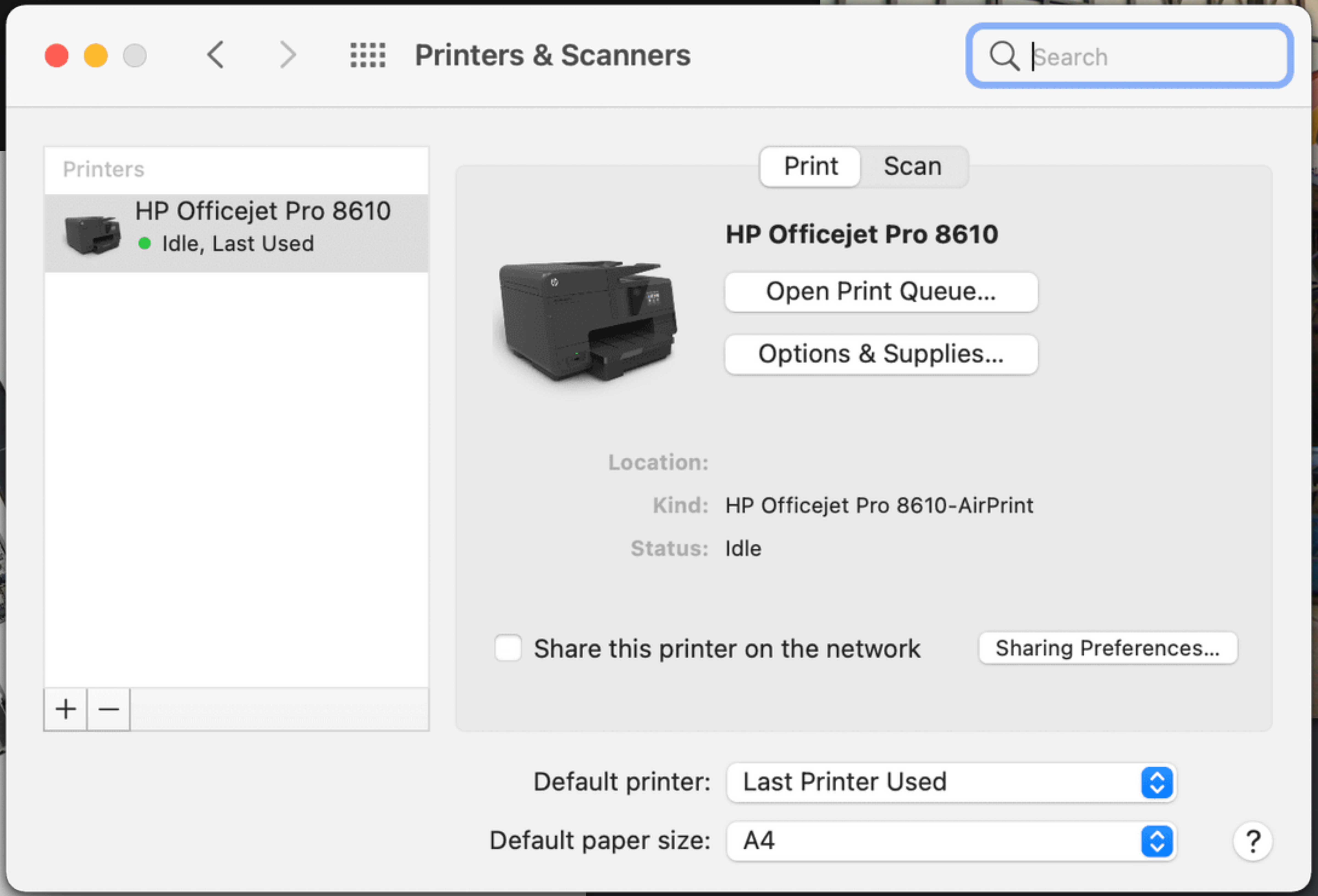


Continuous Delivery ltd

Modularity






Printers & Scanners

Search

Printers

HP Officejet Pro 8610
● Idle, Last Used

+ −

Print | Scan

HP Officejet Pro 8610

Open Print Queue...

Options & Supplies...

Location:
Kind: HP Officejet Pro 8610-AirPrint
Status: Idle

☐ Share this printer on the network

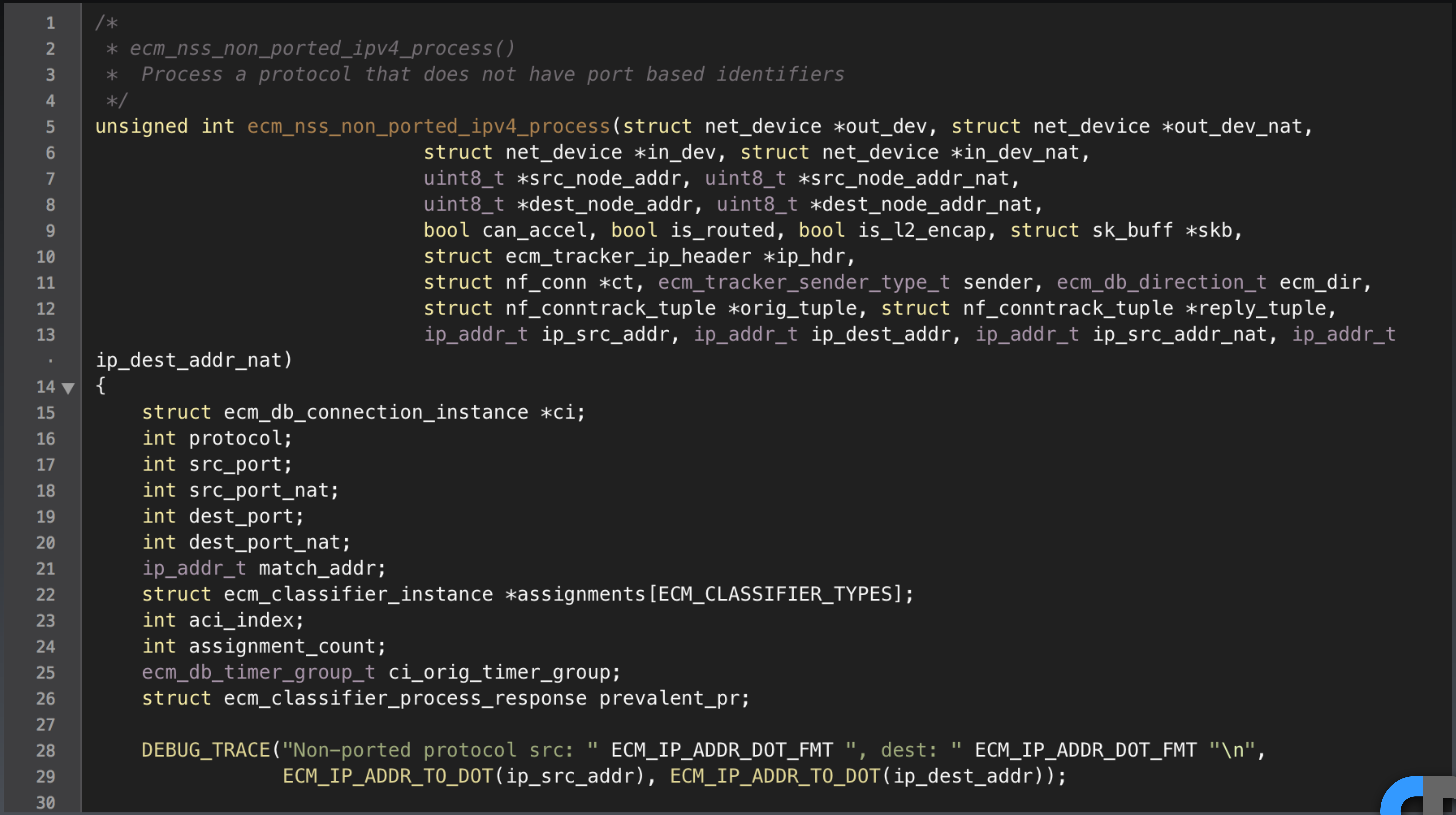
Sharing Preferences...

Default printer: Last Printer Used

Default paper size: A4

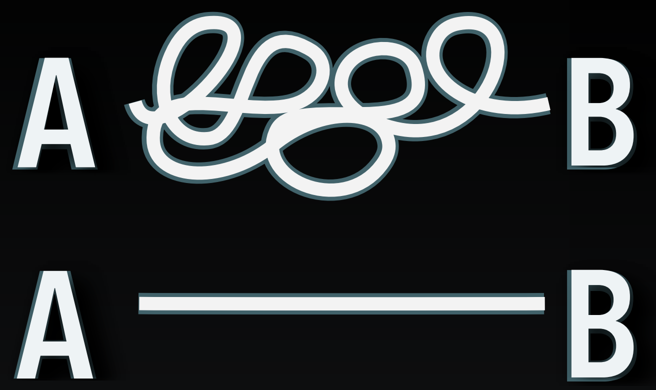
?

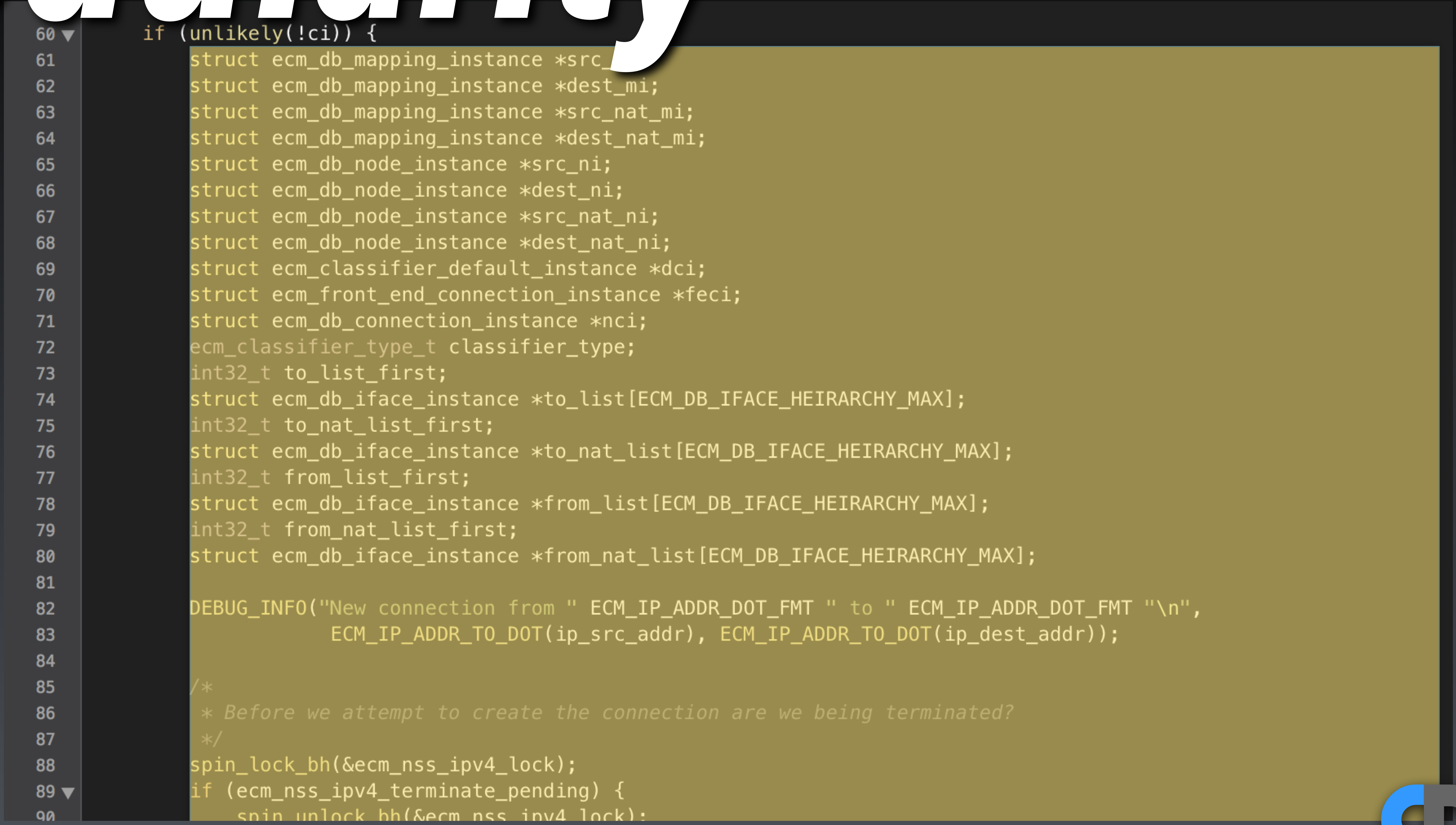
Continuous Delivery ltd



```c
/*
 * ecm_nss_non_ported_ipv4_process()
 *  Process a protocol that does not have port based identifiers
 */
unsigned int ecm_nss_non_ported_ipv4_process(struct net_device *out_dev, struct net_device *out_dev_nat,
                            struct net_device *in_dev, struct net_device *in_dev_nat,
                            uint8_t *src_node_addr, uint8_t *src_node_addr_nat,
                            uint8_t *dest_node_addr, uint8_t *dest_node_addr_nat,
                            bool can_accel, bool is_routed, bool is_l2_encap, struct sk_buff *skb,
                            struct ecm_tracker_ip_header *ip_hdr,
                            struct nf_conn *ct, ecm_tracker_sender_type_t sender, ecm_db_direction_t ecm_dir,
                            struct nf_conntrack_tuple *orig_tuple, struct nf_conntrack_tuple *reply_tuple,
                            ip_addr_t ip_src_addr, ip_addr_t ip_dest_addr, ip_addr_t ip_src_addr_nat, ip_addr_t ip_dest_addr_nat)
{
    struct ecm_db_connection_instance *ci;
    int protocol;
    int src_port;
    int src_port_nat;
    int dest_port;
    int dest_port_nat;
    ip_addr_t match_addr;
    struct ecm_classifier_instance *assignments[ECM_CLASSIFIER_TYPES];
    int aci_index;
    int assignment_count;
    ecm_db_timer_group_t ci_orig_timer_group;
    struct ecm_classifier_process_response prevalent_pr;

    DEBUG_TRACE("Non-ported protocol src: " ECM_IP_ADDR_DOT_FMT ", dest: " ECM_IP_ADDR_DOT_FMT "\n",
            ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));

```

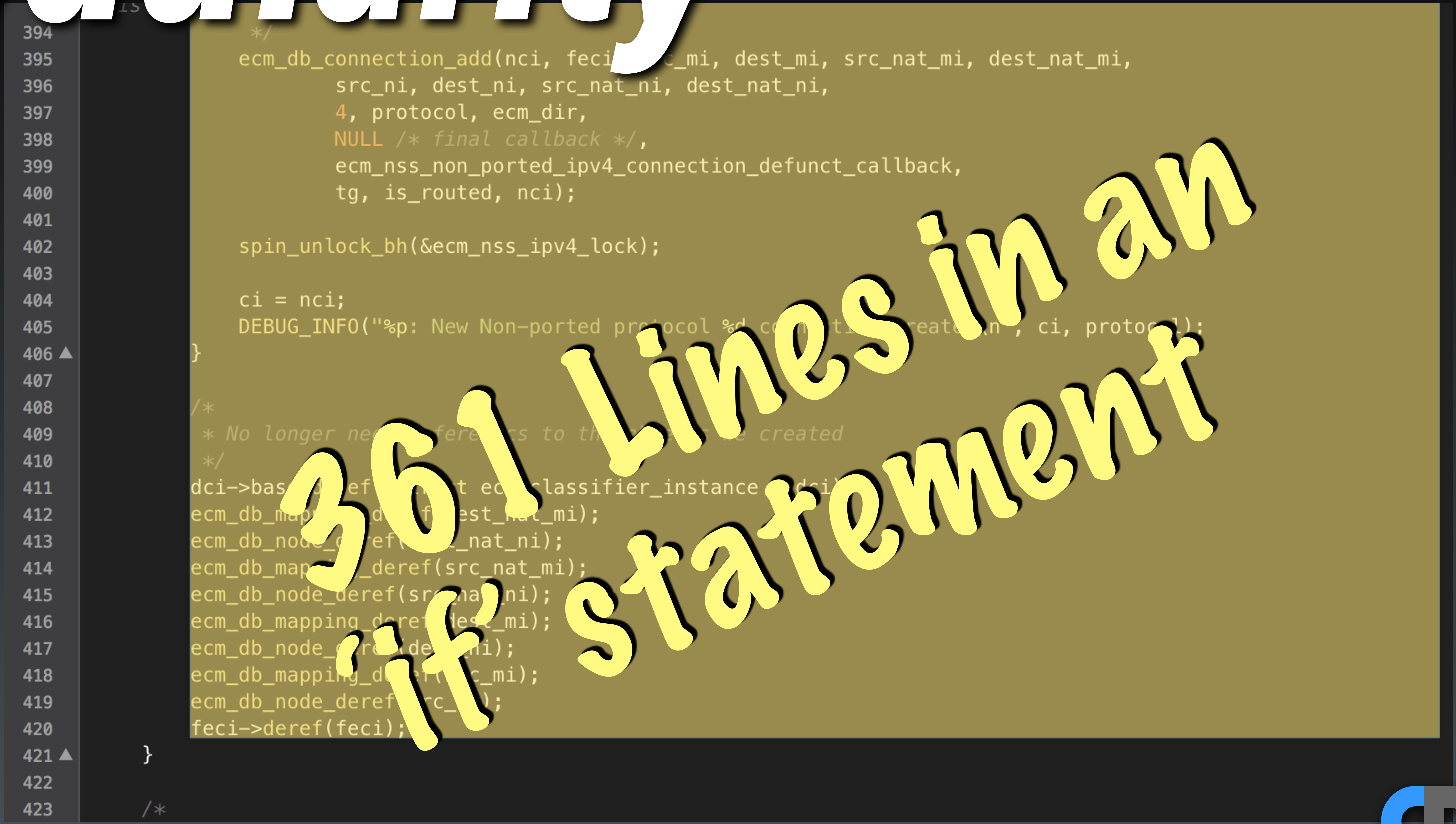
Modularity




```
    if (unlikely(!ci)) {
        struct ecm_db_mapping_instance *src_
        struct ecm_db_mapping_instance *dest_mi;
        struct ecm_db_mapping_instance *src_nat_mi;
        struct ecm_db_mapping_instance *dest_nat_mi;
        struct ecm_db_node_instance *src_ni;
        struct ecm_db_node_instance *dest_ni;
        struct ecm_db_node_instance *src_nat_ni;
        struct ecm_db_node_instance *dest_nat_ni;
        struct ecm_classifier_default_instance *dci;
        struct ecm_front_end_connection_instance *feci;
        struct ecm_db_connection_instance *nci;
        ecm_classifier_type_t classifier_type;
        int32_t to_list_first;
        struct ecm_db_iface_instance *to_list[ECM_DB_IFACE_HEIRARCHY_MAX];
        int32_t to_nat_list_first;
        struct ecm_db_iface_instance *to_nat_list[ECM_DB_IFACE_HEIRARCHY_MAX];
        int32_t from_list_first;
        struct ecm_db_iface_instance *from_list[ECM_DB_IFACE_HEIRARCHY_MAX];
        int32_t from_nat_list_first;
        struct ecm_db_iface_instance *from_nat_list[ECM_DB_IFACE_HEIRARCHY_MAX];

        DEBUG_INFO("New connection from " ECM_IP_ADDR_DOT_FMT " to " ECM_IP_ADDR_DOT_FMT "\n",
                    ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));

        /*
         * Before we attempt to create the connection are we being terminated?
         */
        spin_lock_bh(&ecm_nss_ipv4_lock);
        if (ecm_nss_ipv4_terminate_pending) {
            spin_unlock_bh(&ecm_nss_ipv4_lock);
```

Continuous Delivery ltd

Modularity

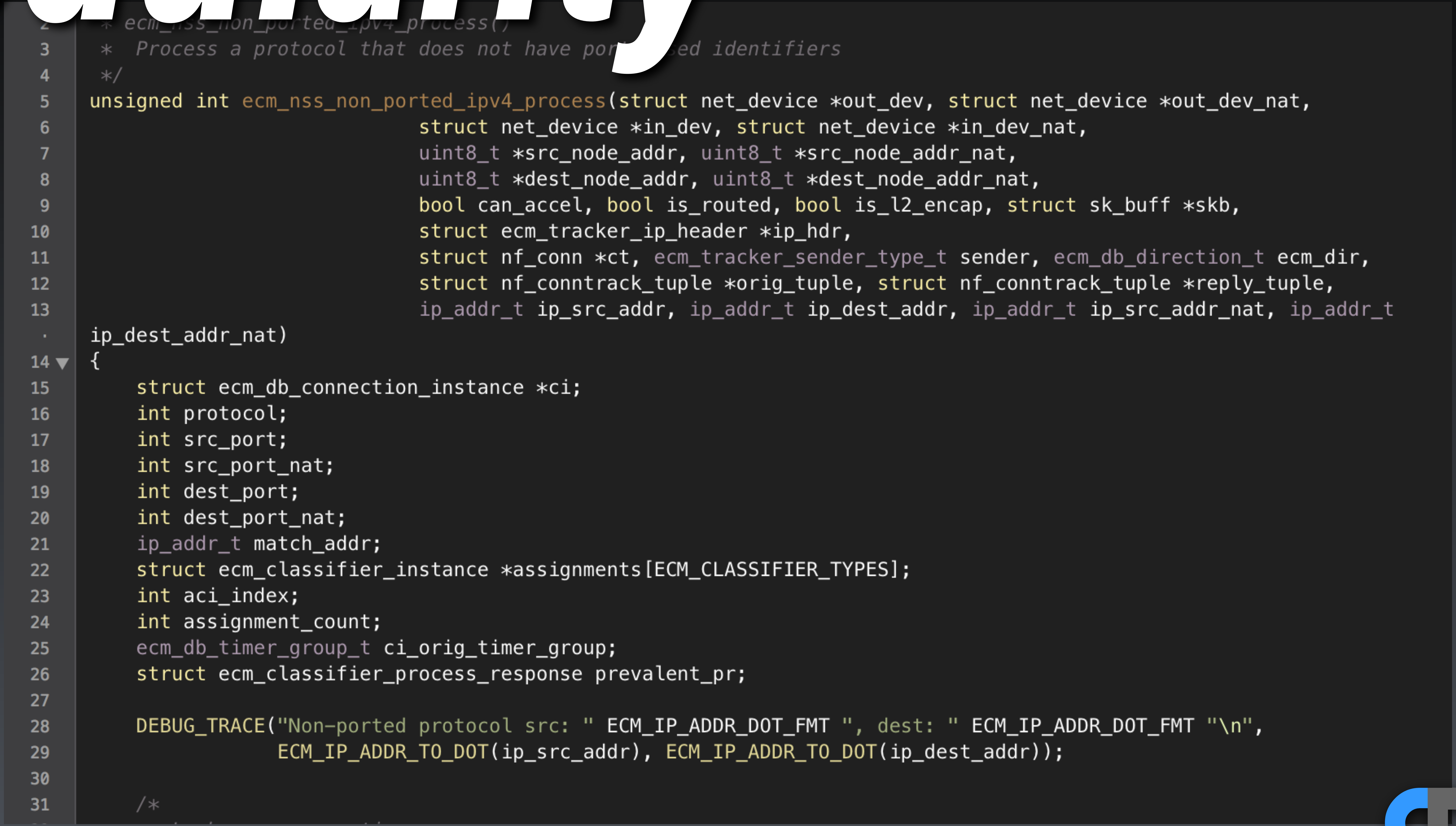


```
         */
        ecm_db_connection_add(nci, feci, src_mi, dest_mi, src_nat_mi, dest_nat_mi,
                src_ni, dest_ni, src_nat_ni, dest_nat_ni,
                4, protocol, ecm_dir,
                NULL /* final callback */,
                ecm_nss_non_ported_ipv4_connection_defunct_callback,
                tg, is_routed, nci);

        spin_unlock_bh(&ecm_nss_ipv4_lock);

        ci = nci;
        DEBUG_INFO("%p: New Non-ported protocol %d connection created\n", ci, protocol);
    }

    /*
     * No longer need references to the objects we created
     */
    dci->base.deref((struct ecm_classifier_instance *)dci);
    ecm_db_mapping_deref(dest_nat_mi);
    ecm_db_node_deref(dest_nat_ni);
    ecm_db_mapping_deref(src_nat_mi);
    ecm_db_node_deref(src_nat_ni);
    ecm_db_mapping_deref(dest_mi);
    ecm_db_node_deref(dest_ni);
    ecm_db_mapping_deref(src_mi);
    ecm_db_node_deref(src_ni);
    feci->deref(feci);
}

/*
```

361 Lines in an 'if' statement

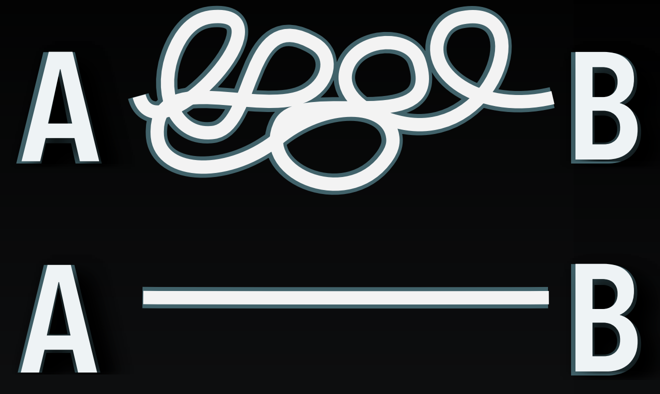
Continuous Delivery ltd

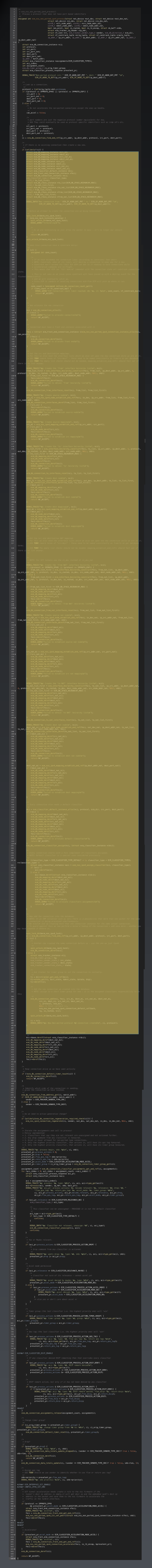
Modularity


```
 * ecm_nss_non_ported_ipv4_process()
 *  Process a protocol that does not have por[illegible]ed identifiers
 */
unsigned int ecm_nss_non_ported_ipv4_process(struct net_device *out_dev, struct net_device *out_dev_nat,
                            struct net_device *in_dev, struct net_device *in_dev_nat,
                            uint8_t *src_node_addr, uint8_t *src_node_addr_nat,
                            uint8_t *dest_node_addr, uint8_t *dest_node_addr_nat,
                            bool can_accel, bool is_routed, bool is_l2_encap, struct sk_buff *skb,
                            struct ecm_tracker_ip_header *ip_hdr,
                            struct nf_conn *ct, ecm_tracker_sender_type_t sender, ecm_db_direction_t ecm_dir,
                            struct nf_conntrack_tuple *orig_tuple, struct nf_conntrack_tuple *reply_tuple,
                            ip_addr_t ip_src_addr, ip_addr_t ip_dest_addr, ip_addr_t ip_src_addr_nat, ip_addr_t ip_dest_addr_nat)
{
    struct ecm_db_connection_instance *ci;
    int protocol;
    int src_port;
    int src_port_nat;
    int dest_port;
    int dest_port_nat;
    ip_addr_t match_addr;
    struct ecm_classifier_instance *assignments[ECM_CLASSIFIER_TYPES];
    int aci_index;
    int assignment_count;
    ecm_db_timer_group_t ci_orig_timer_group;
    struct ecm_classifier_process_response prevalent_pr;

    DEBUG_TRACE("Non-ported protocol src: " ECM_IP_ADDR_DOT_FMT ", dest: " ECM_IP_ADDR_DOT_FMT "\n",
                ECM_IP_ADDR_TO_DOT(ip_src_addr), ECM_IP_ADDR_TO_DOT(ip_dest_addr));

    /*
```

Continuous Delivery ltd

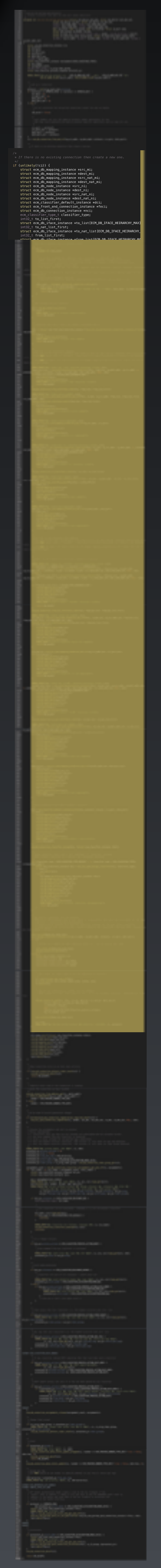
Modularity





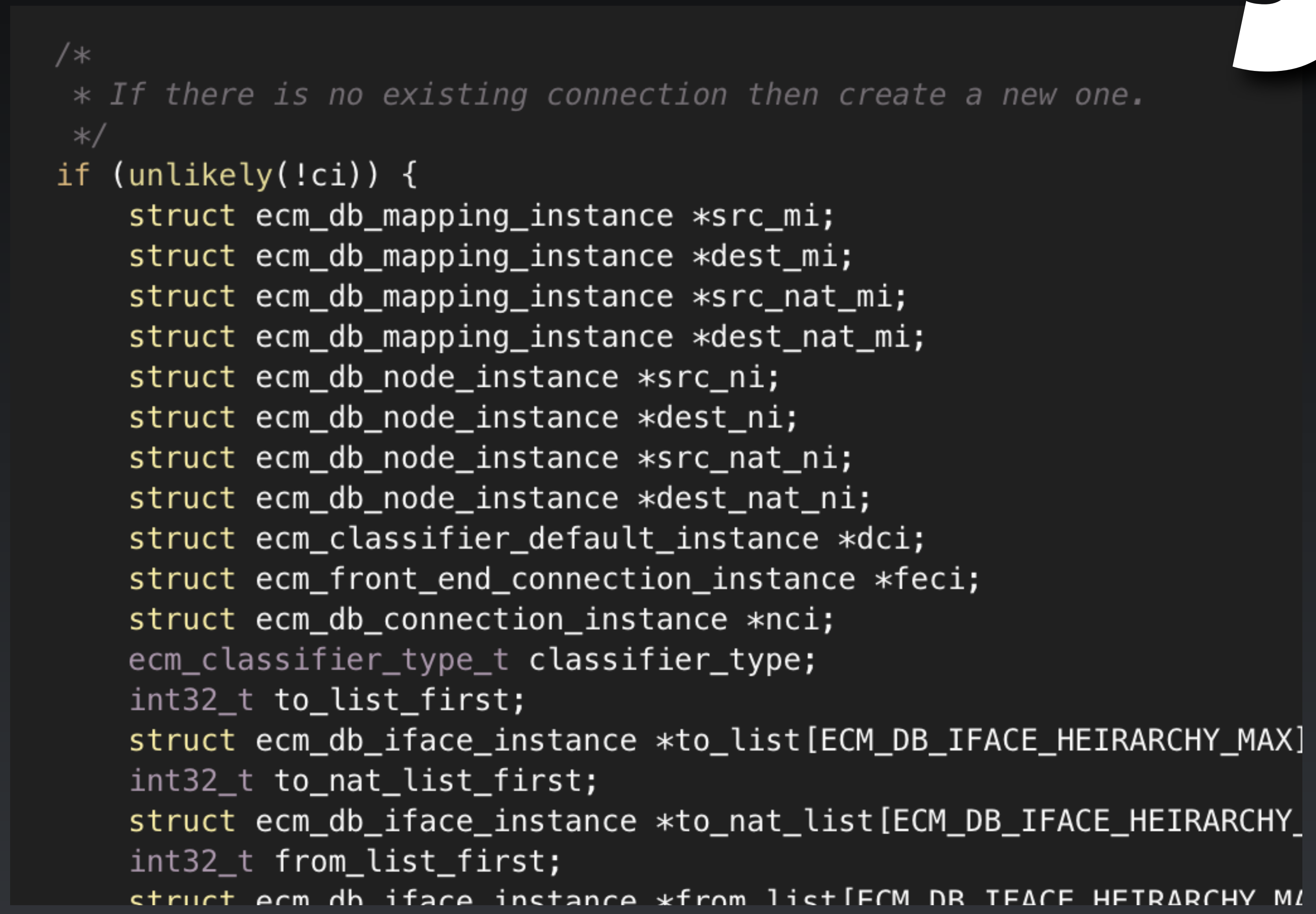
Continuous Delivery ltd

Modularity



Continuous Delivery ltd

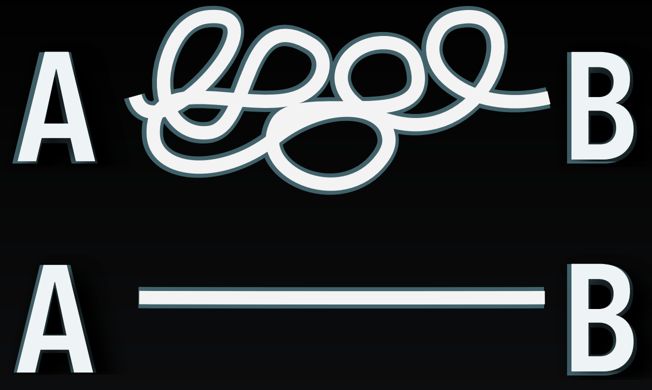
Modularity



```
/*
 * If there is no existing connection then create a new one.
 */
if (unlikely(!ci)) {
	struct ecm_db_mapping_instance *src_mi;
	struct ecm_db_mapping_instance *dest_mi;
	struct ecm_db_mapping_instance *src_nat_mi;
	struct ecm_db_mapping_instance *dest_nat_mi;
	struct ecm_db_node_instance *src_ni;
	struct ecm_db_node_instance *dest_ni;
	struct ecm_db_node_instance *src_nat_ni;
	struct ecm_db_node_instance *dest_nat_ni;
	struct ecm_classifier_default_instance *dci;
	struct ecm_front_end_connection_instance *feci;
	struct ecm_db_connection_instance *nci;
	ecm_classifier_type_t classifier_type;
	int32_t to_list_first;
	struct ecm_db_iface_instance *to_list[ECM_DB_IFACE_HEIRARCHY_MAX]
	int32_t to_nat_list_first;
	struct ecm_db_iface_instance *to_nat_list[ECM_DB_IFACE_HEIRARCHY_
	int32_t from_list_first;
	struct ecm_db_iface_instance *from_list[ECM_DB_IFACE_HEIRARCHY_MA
```

Continuous Delivery ltd

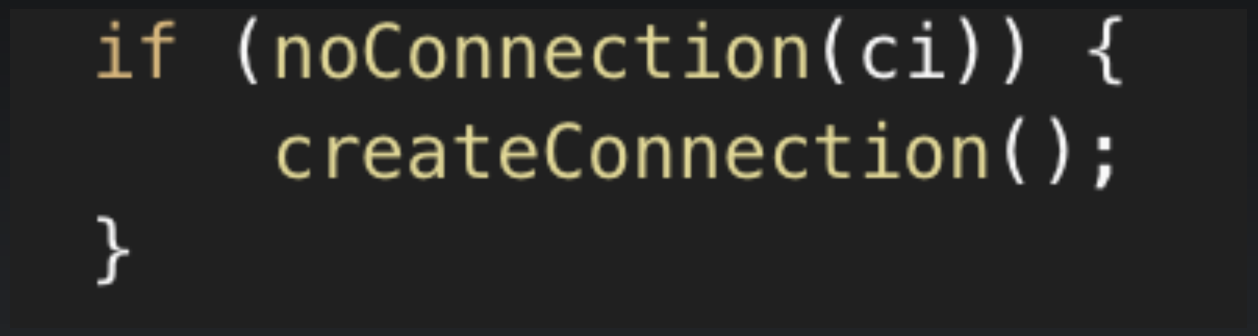
Modularity



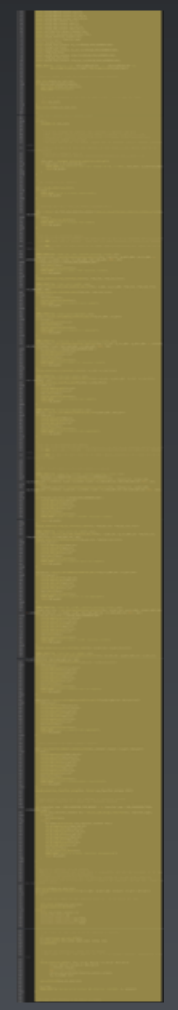
```
if (noConnection(ci)) {
    createConnection();
}
```

Continuous Delivery ltd

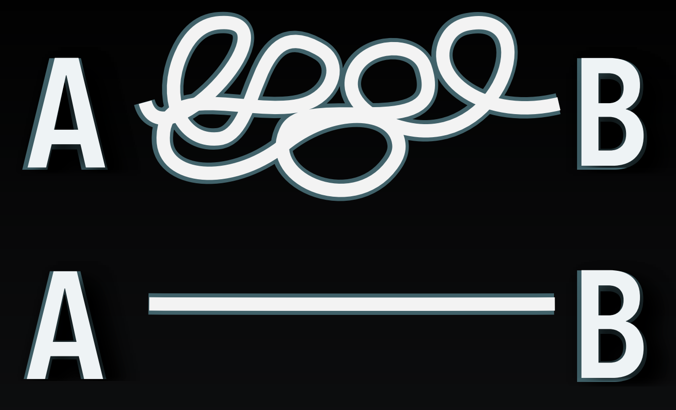
Modularity



```
if (noConnection(ci)) {
    createConnection();
}
```



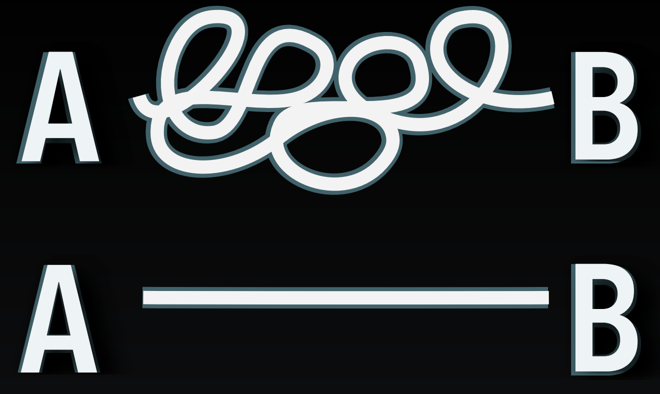
Continuous Delivery ltd



"Good Design is about moving the things that are related closer together, and the things that aren't related further apart" - ***Kent Beck***

Continuous Delivery ltd

"Good Design is about moving the things that are related closer together, and the things that aren't related further apart." - *Kent Beck*



Modularity

things that aren't related further apart

Cohesion

things that are related closer together

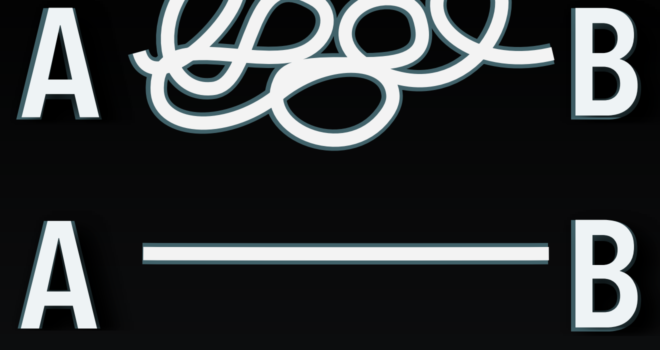
Continuous Delivery ltd

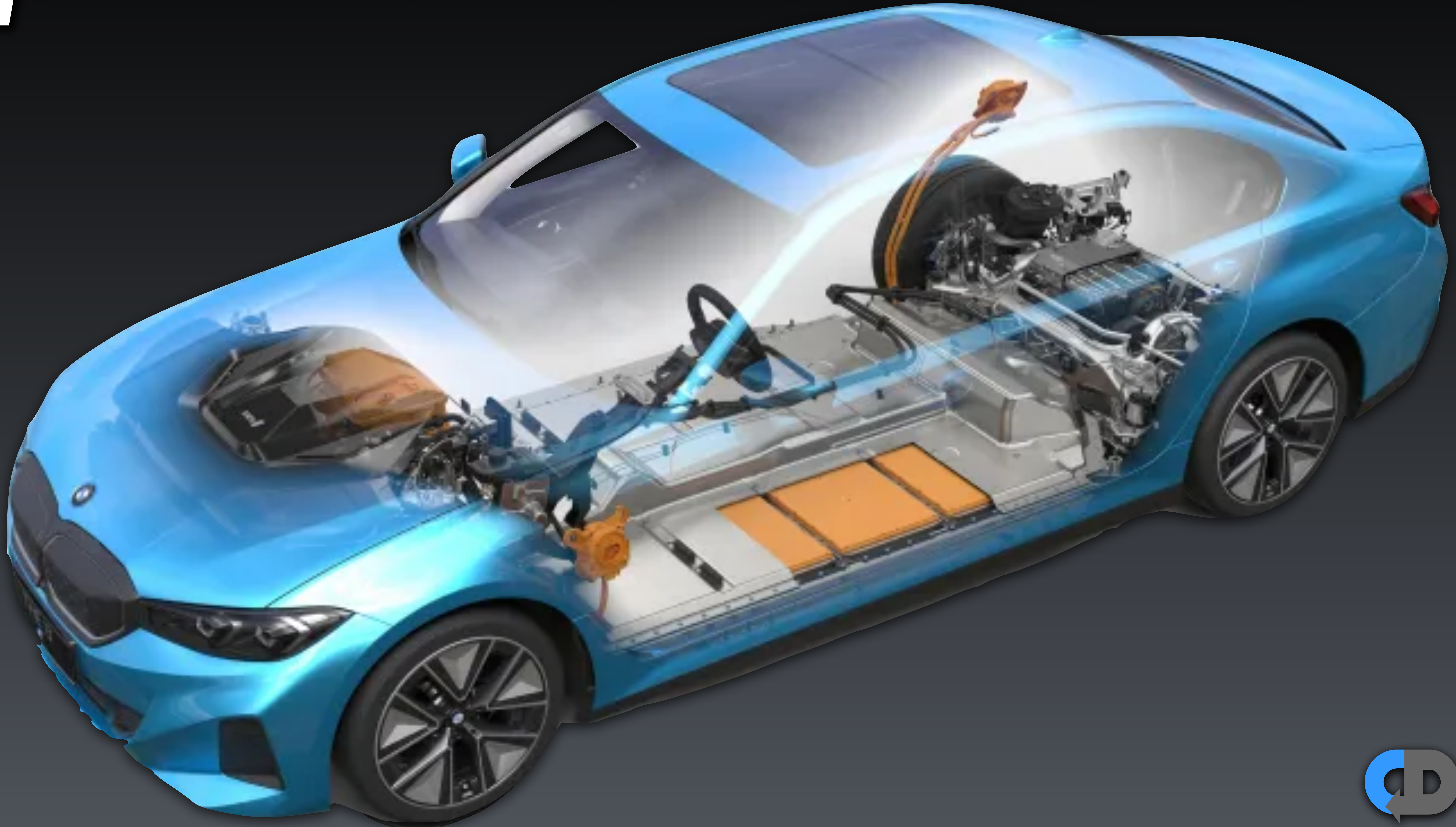
Separation of Concerns

Separation of Concerns

Continuous Delivery ltd

Separation of Concerns



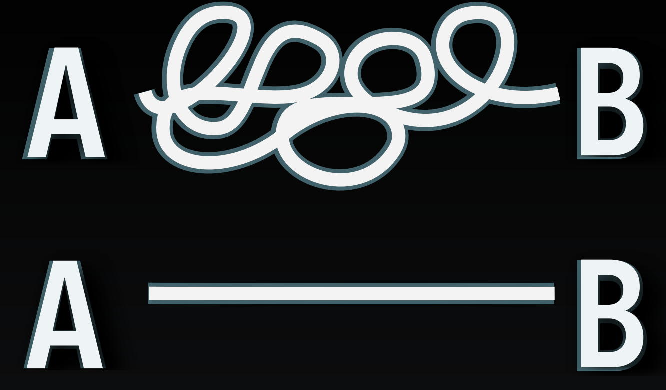


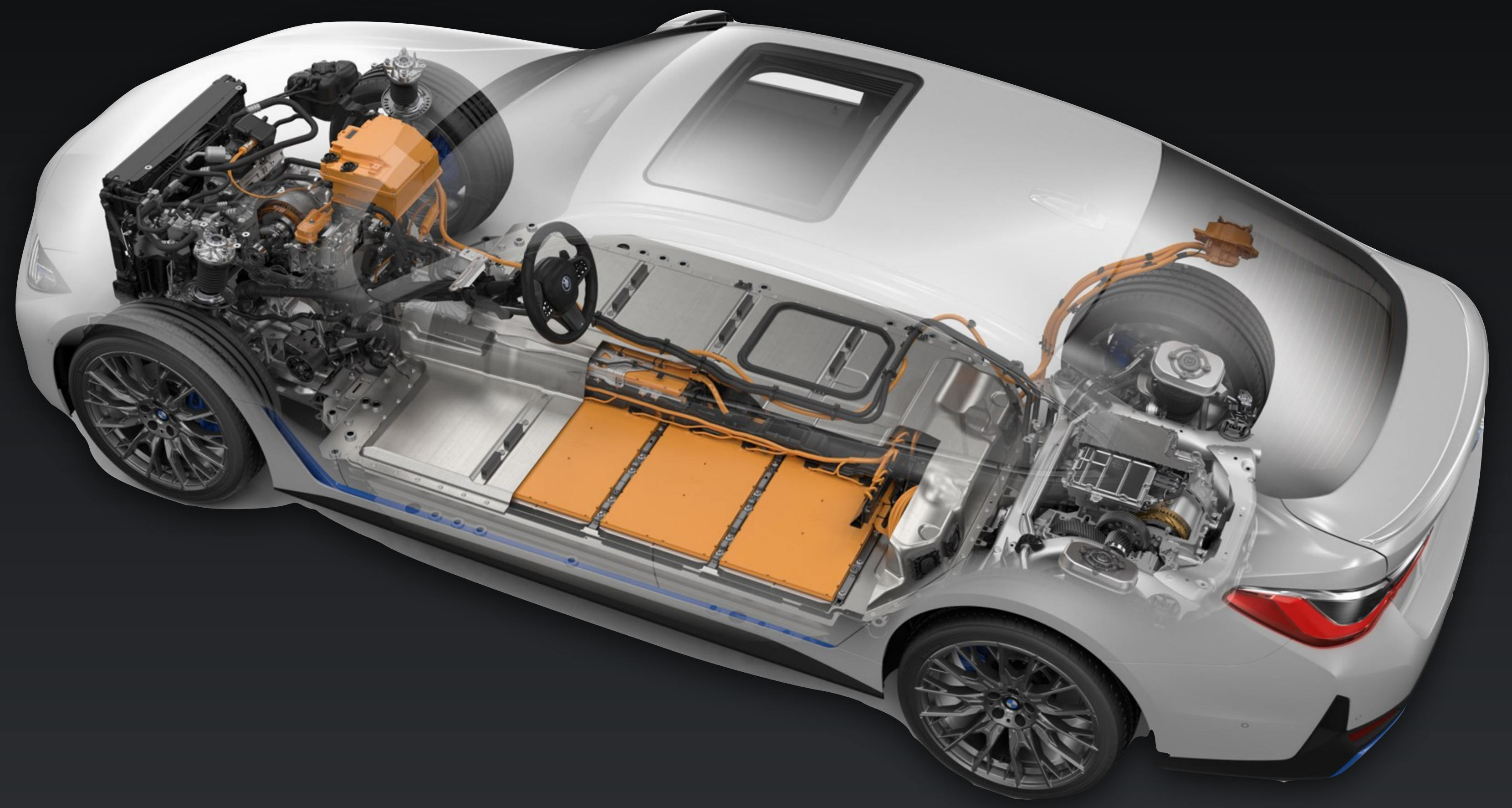
Continuous Delivery ltd

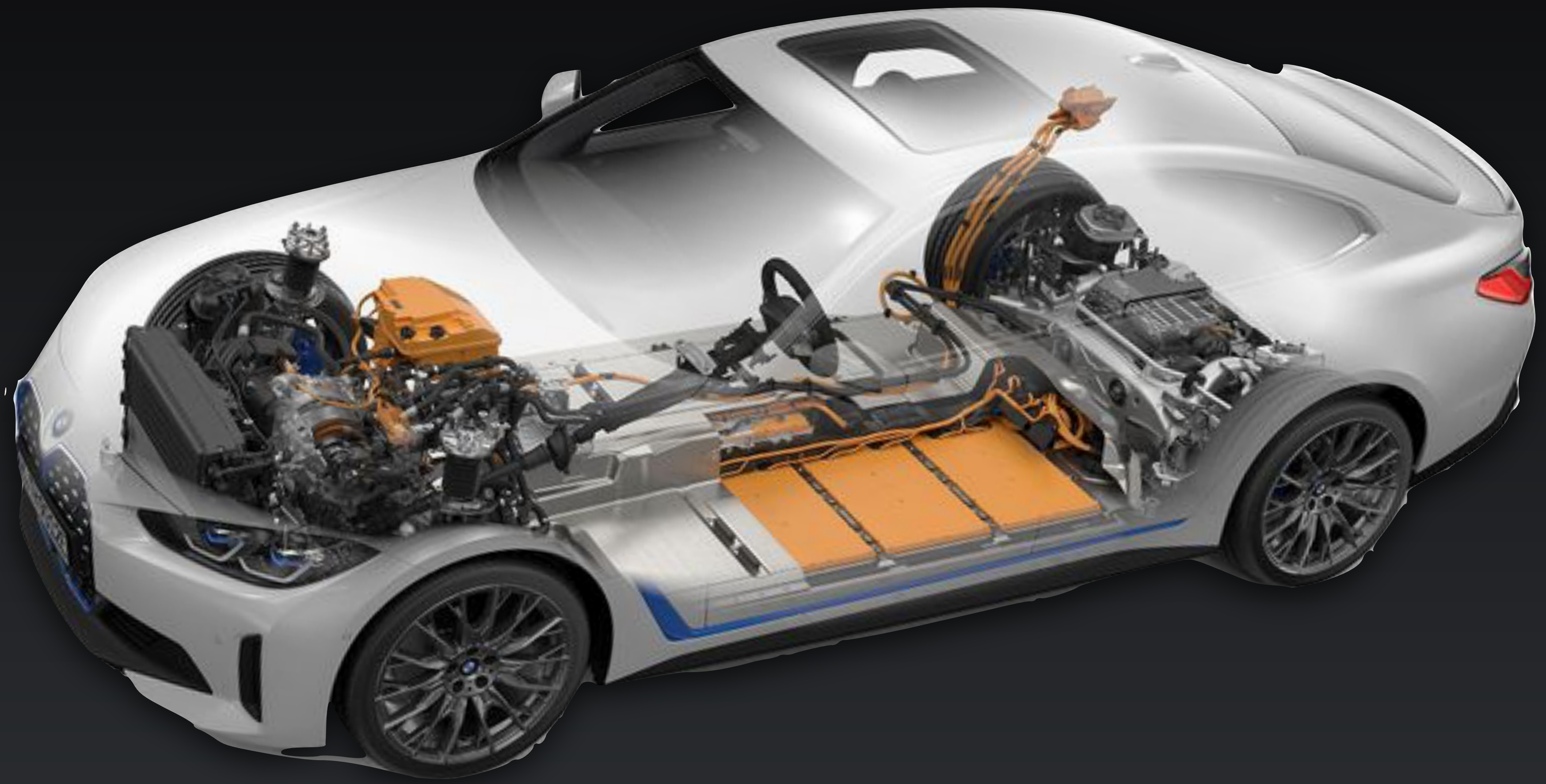
Separation of Concerns

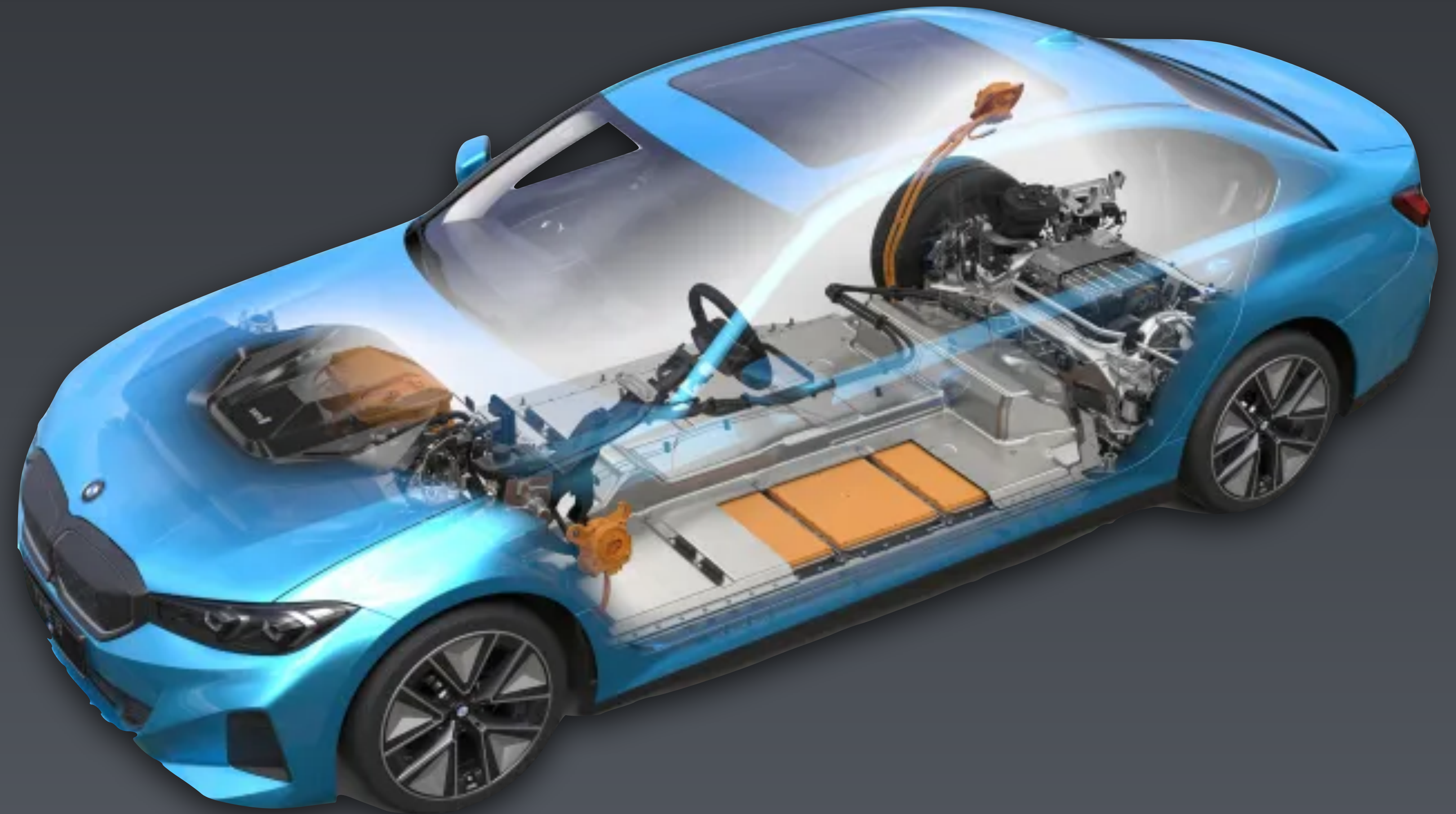
Continuous Delivery ltd

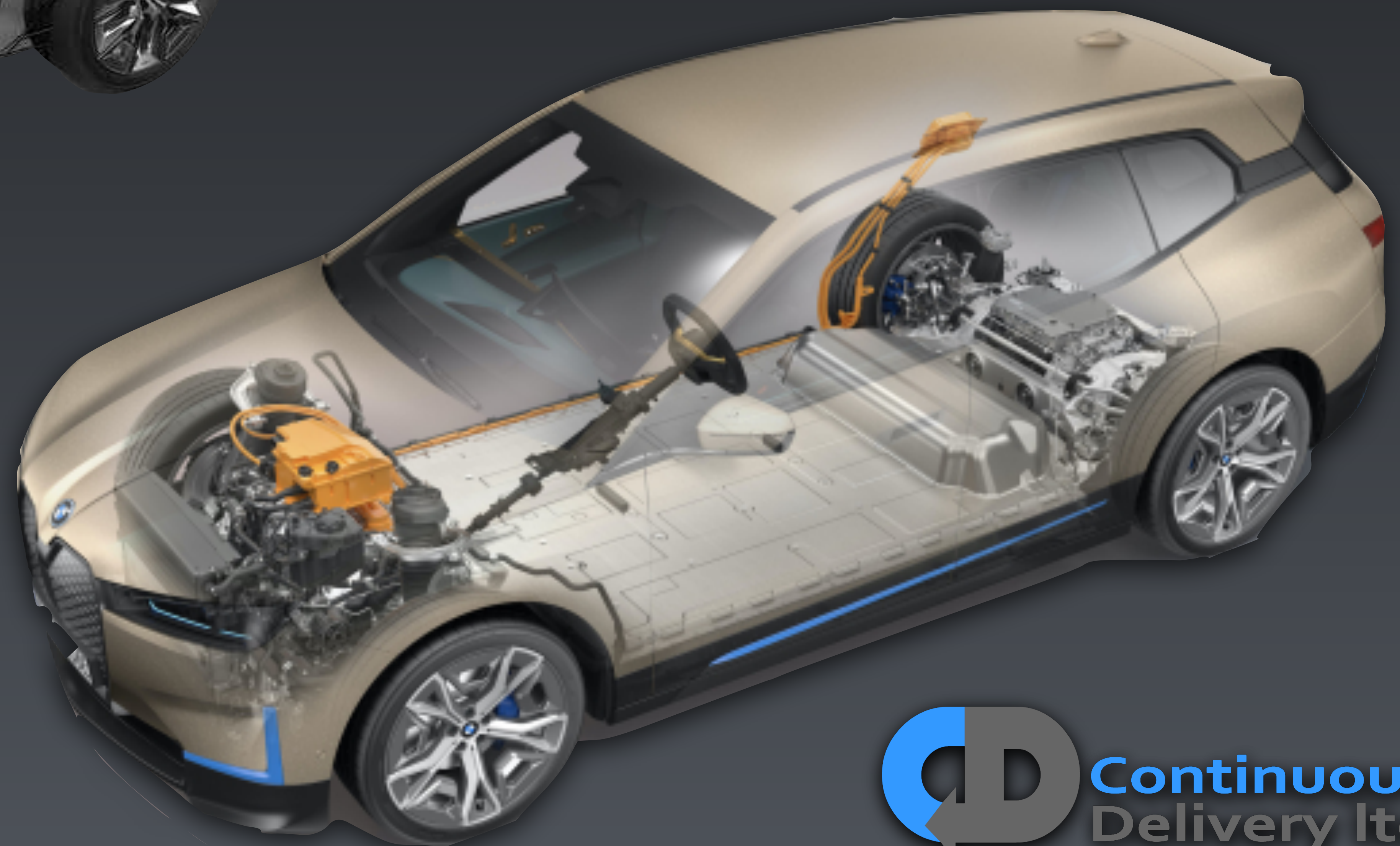
Separation of Concerns









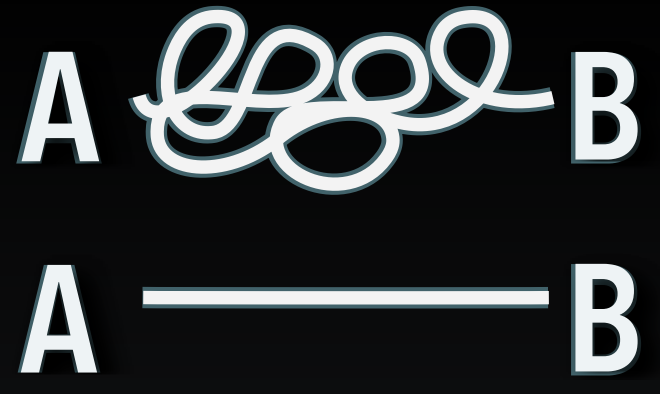


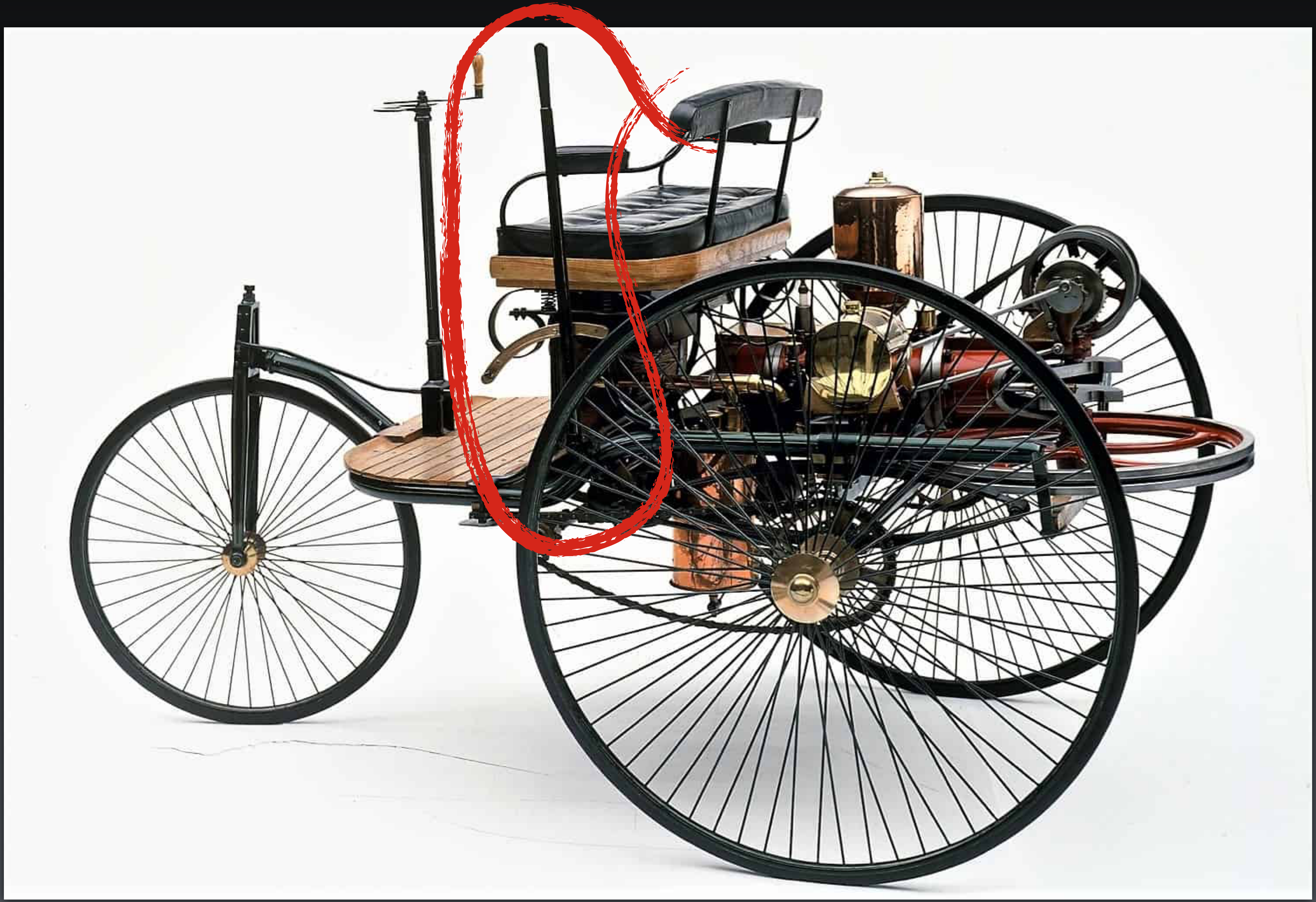
Continuous Delivery ltd

Abstraction

Continuous Delivery ltd

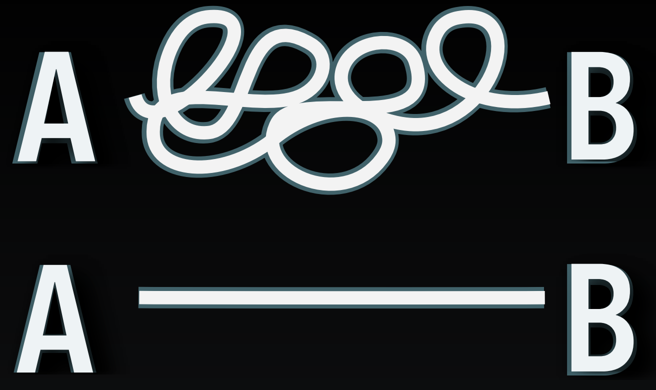
Abstraction

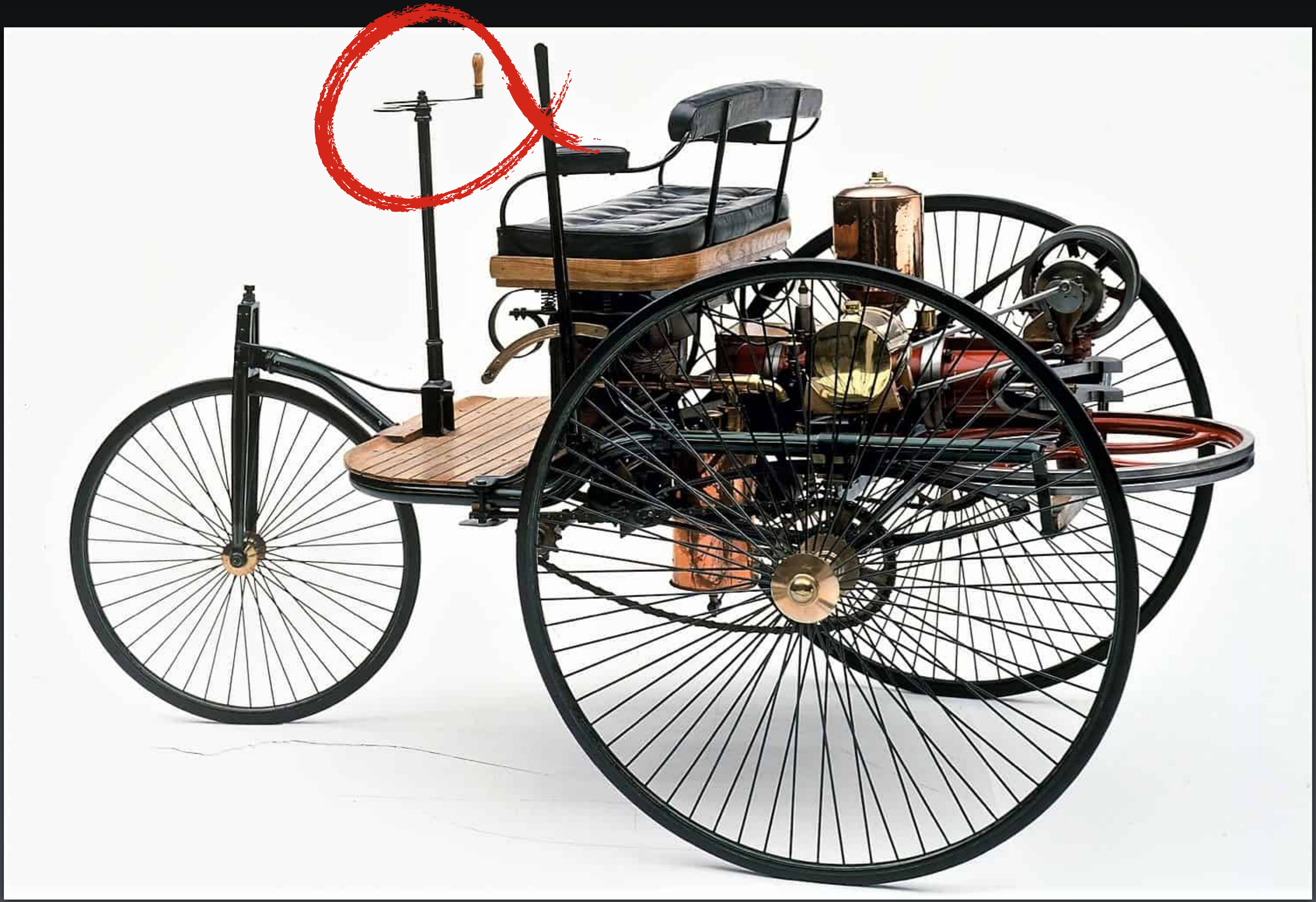




Continuous Delivery ltd

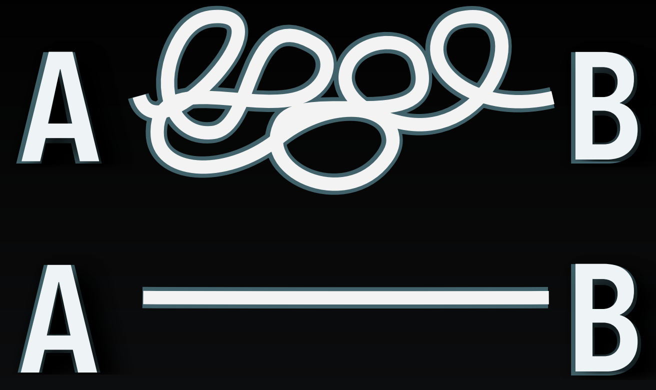
Abstraction





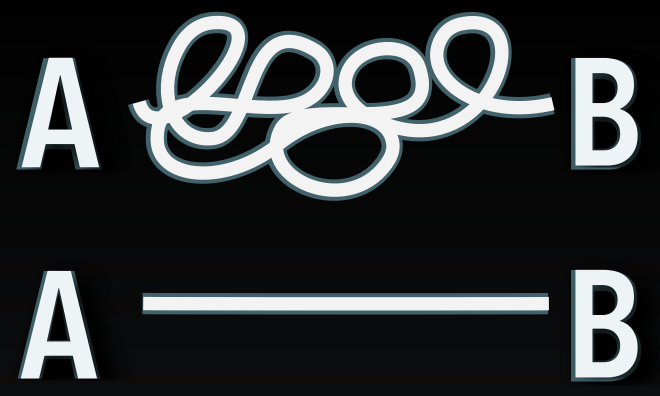
Continuous Delivery ltd

Abstraction

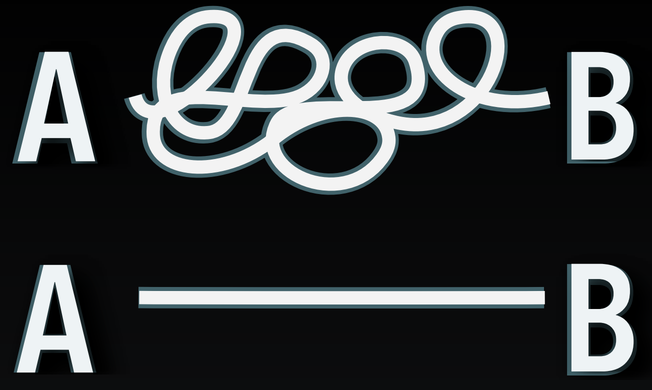


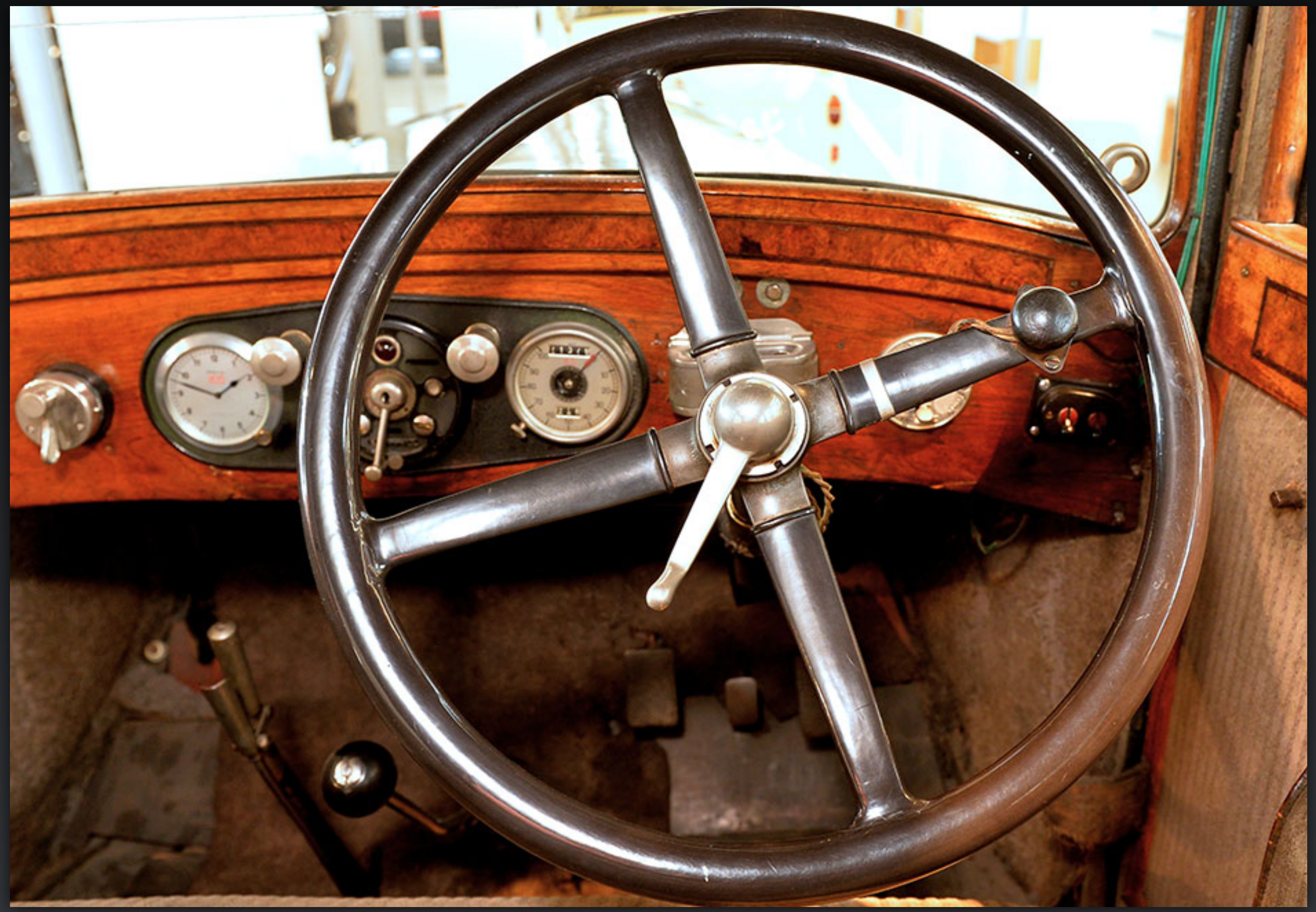
Continuous Delivery ltd

Abstraction

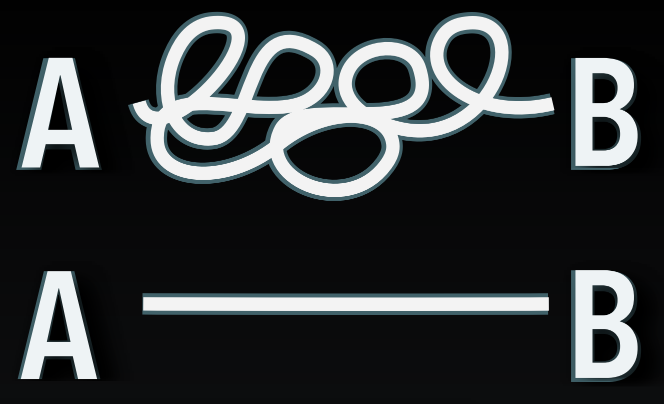


Abstraction





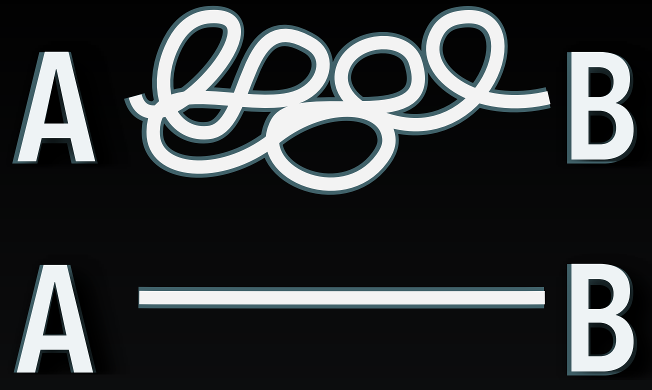
Abstraction

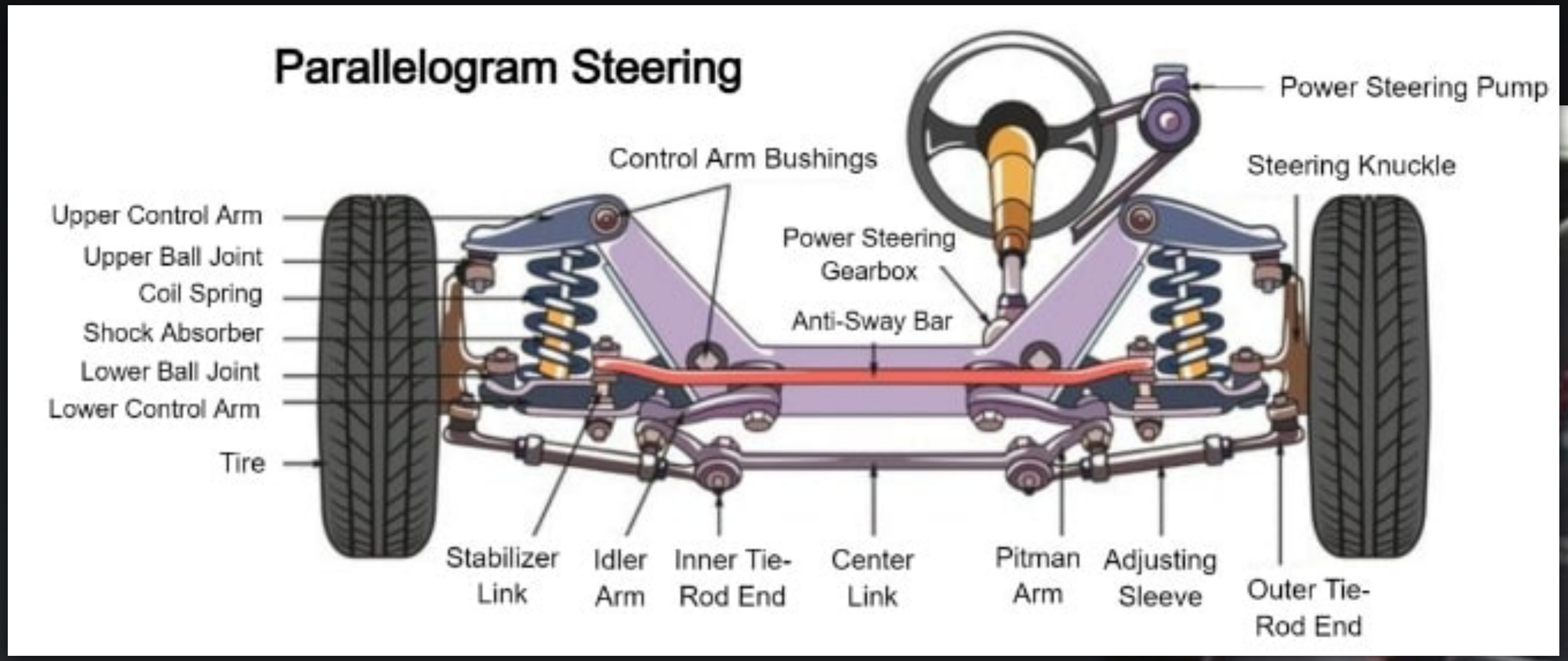


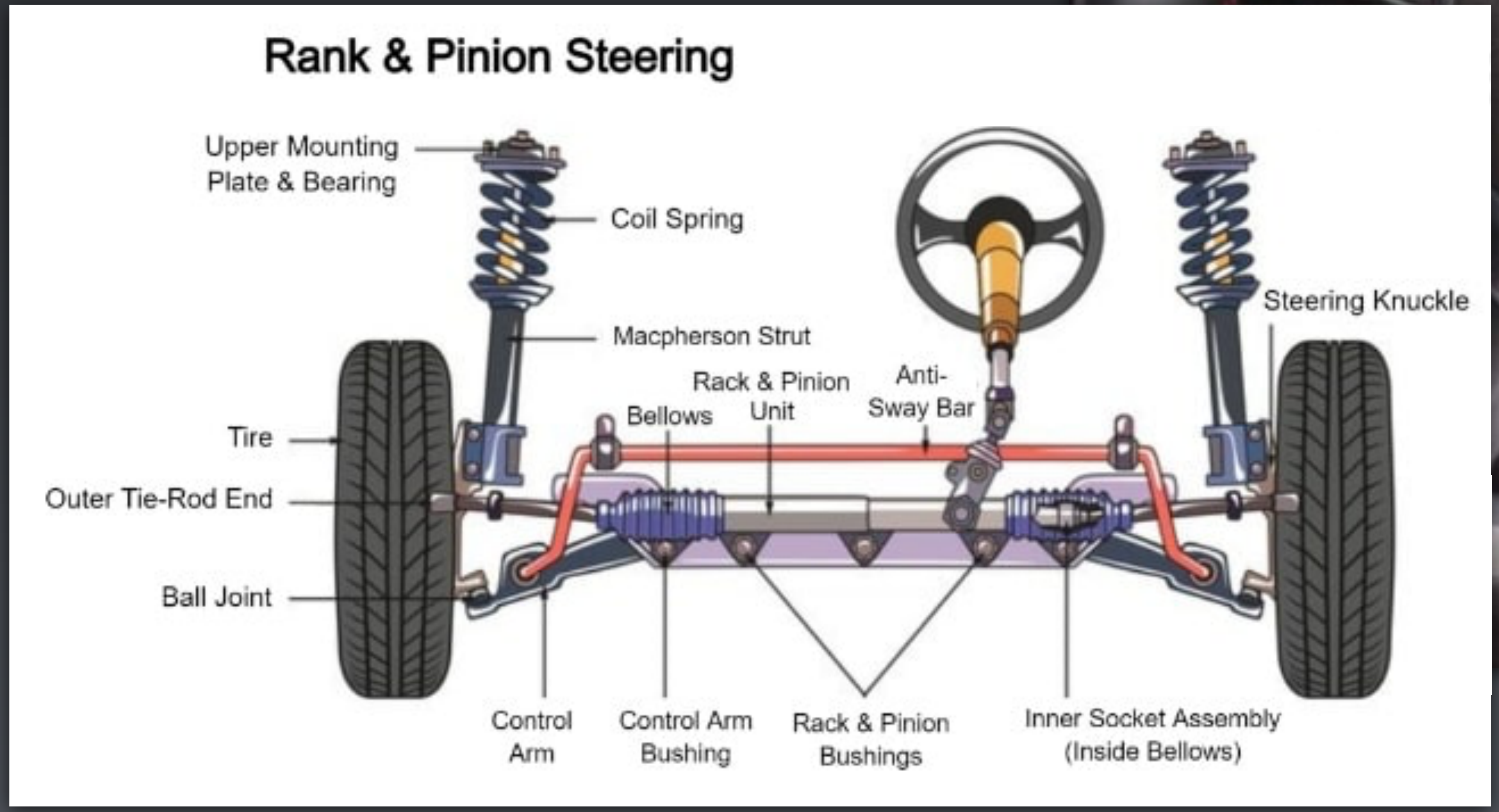
Abstraction

Source: engineeringlearn.com

Abstraction



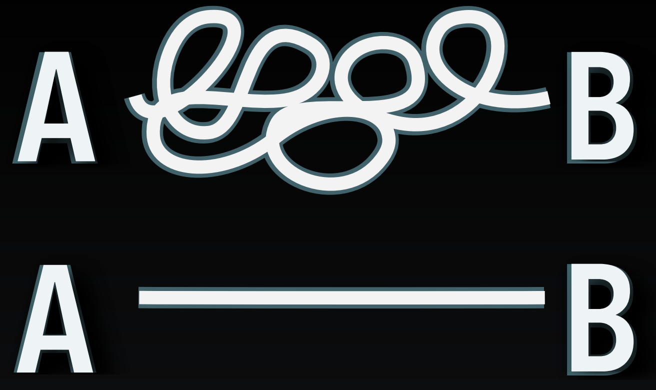


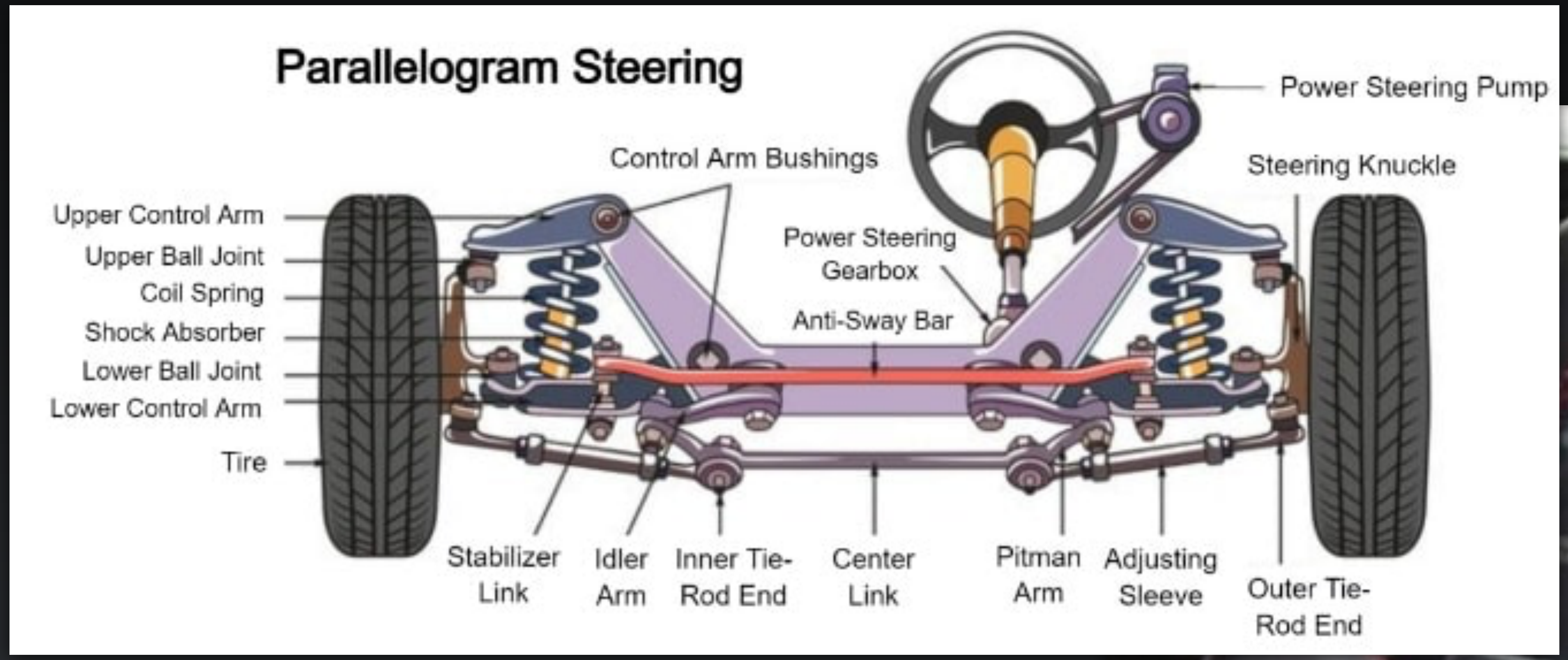


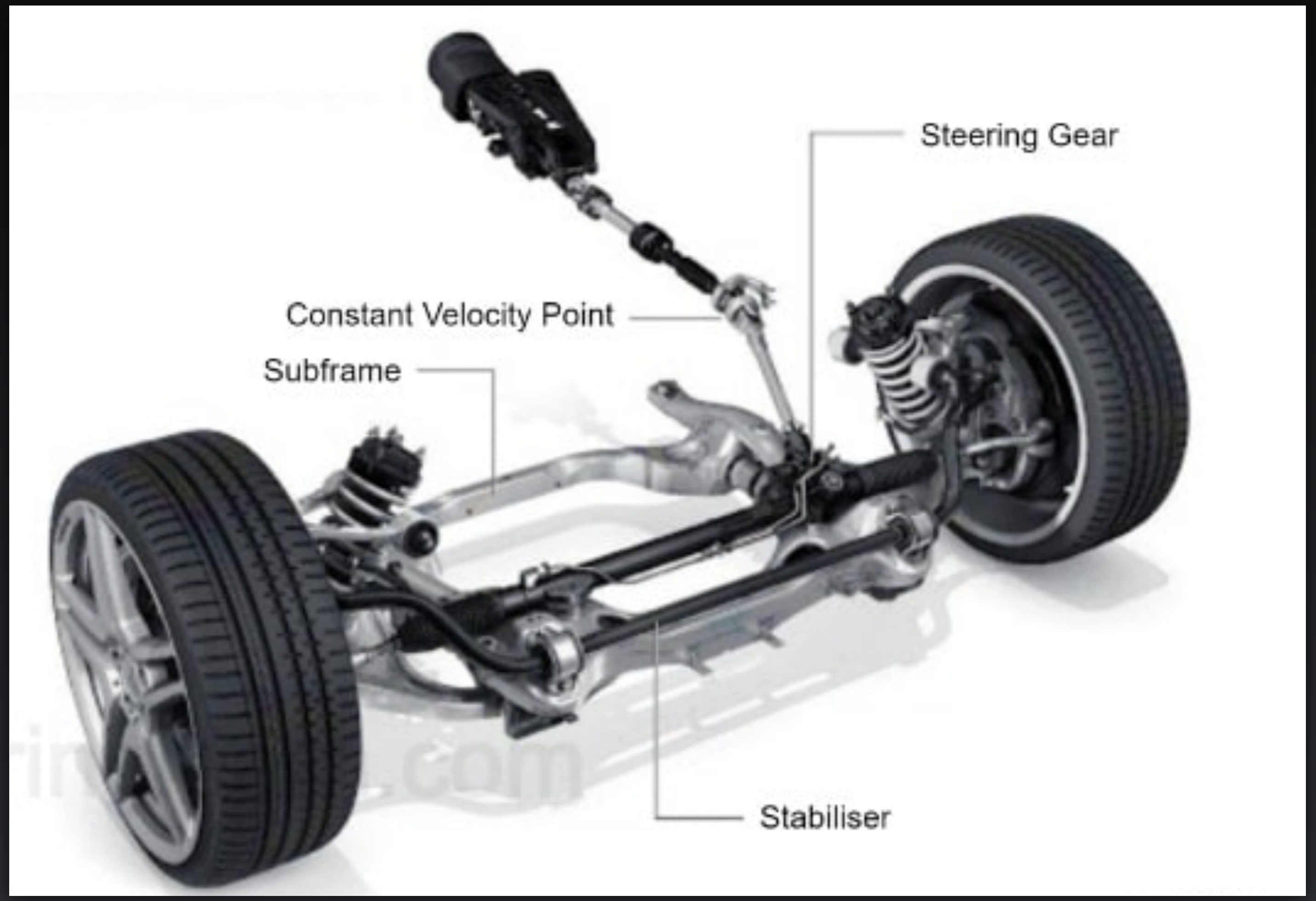
Source: engineeringlearn.com

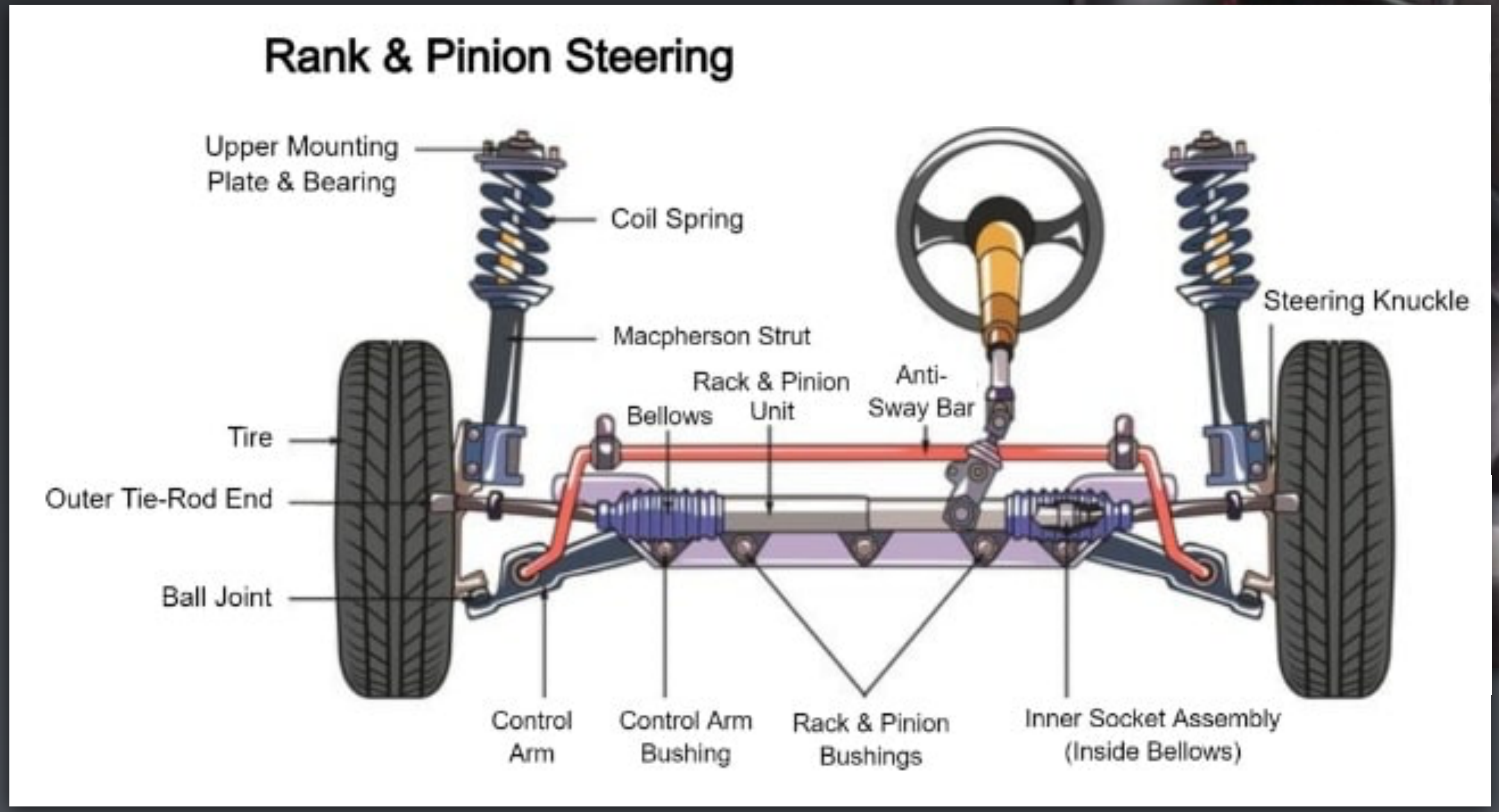
Continuous Delivery ltd

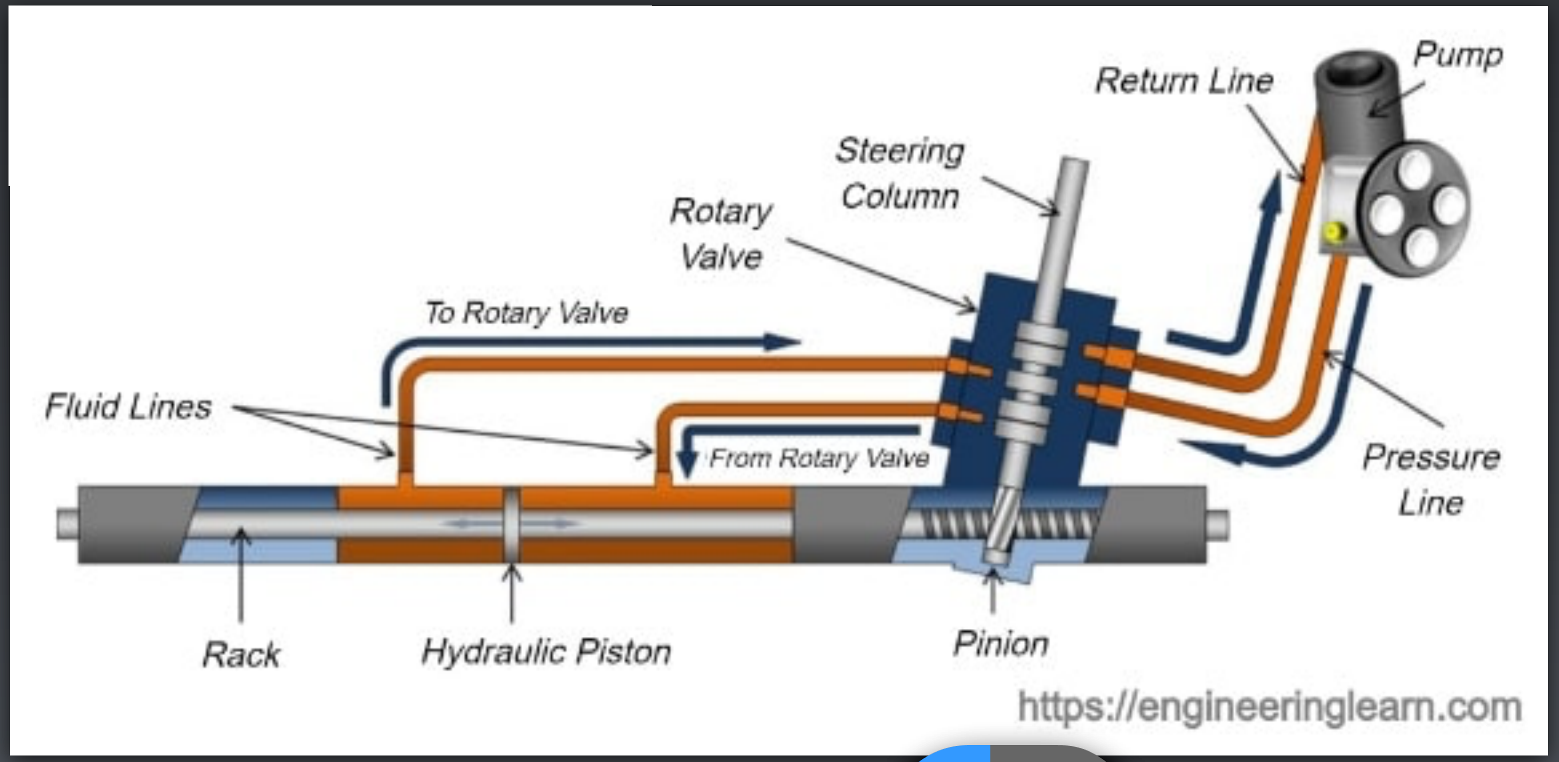
Abstraction











Continuous Delivery ltd

Source: engineeringlearn.com

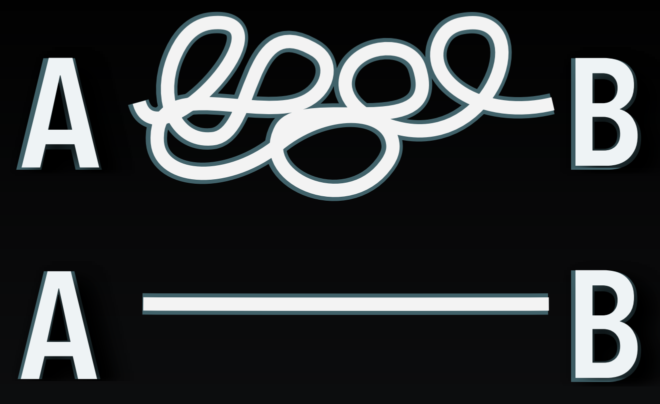
Abstraction

A B

A B

Continuous Delivery ltd

Abstraction

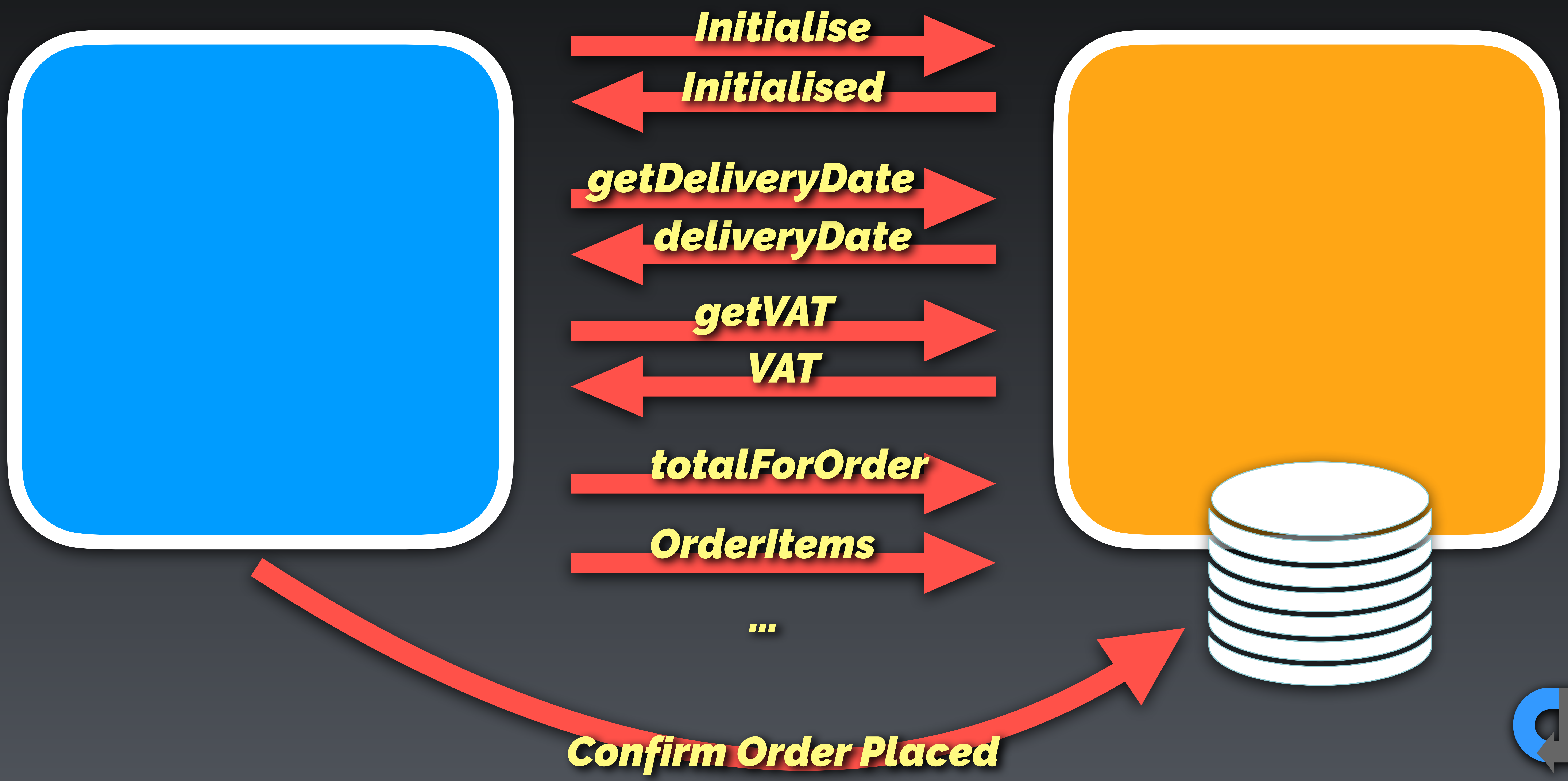


Continuous Delivery ltd

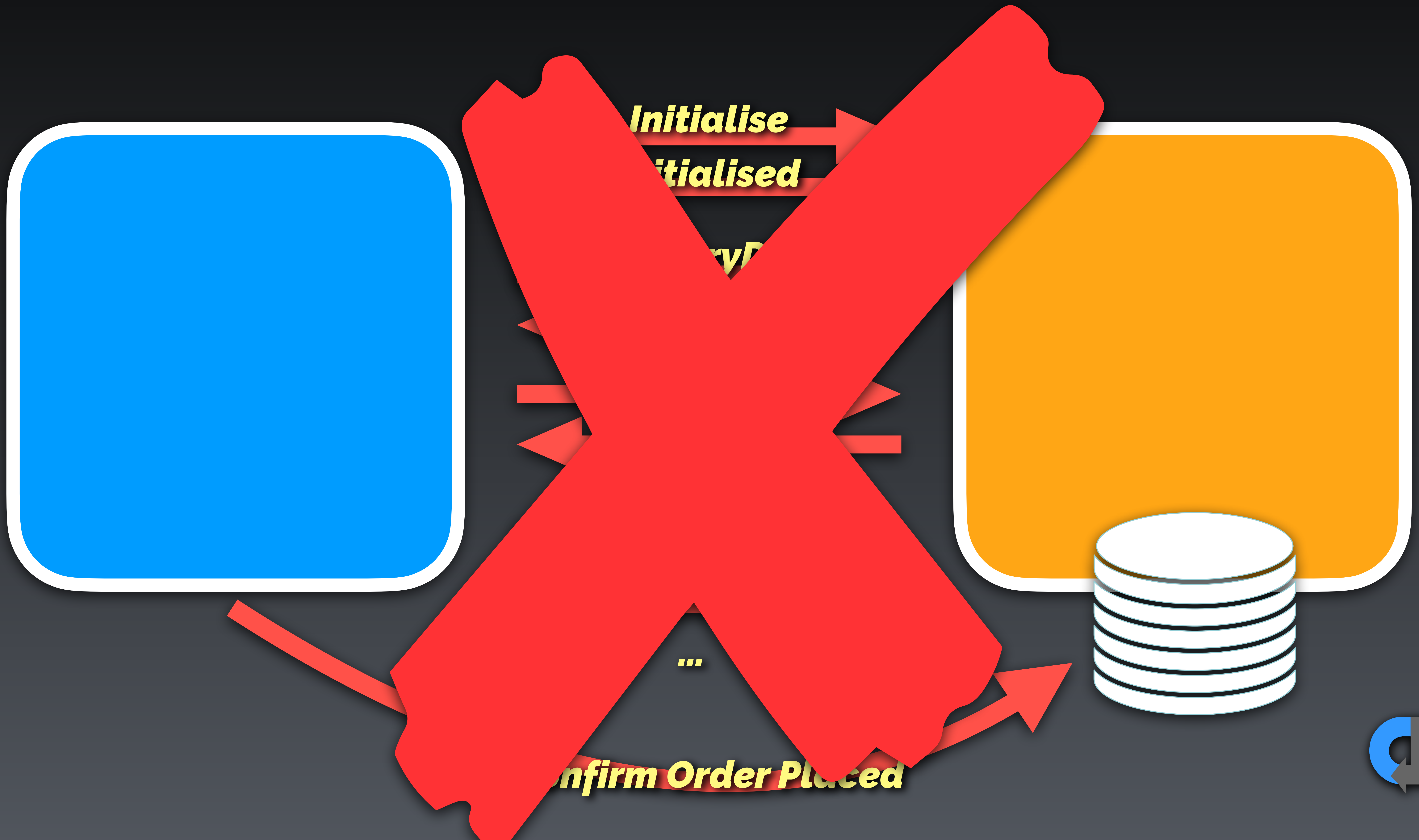
Abstraction

Continuous Delivery ltd

Abstraction

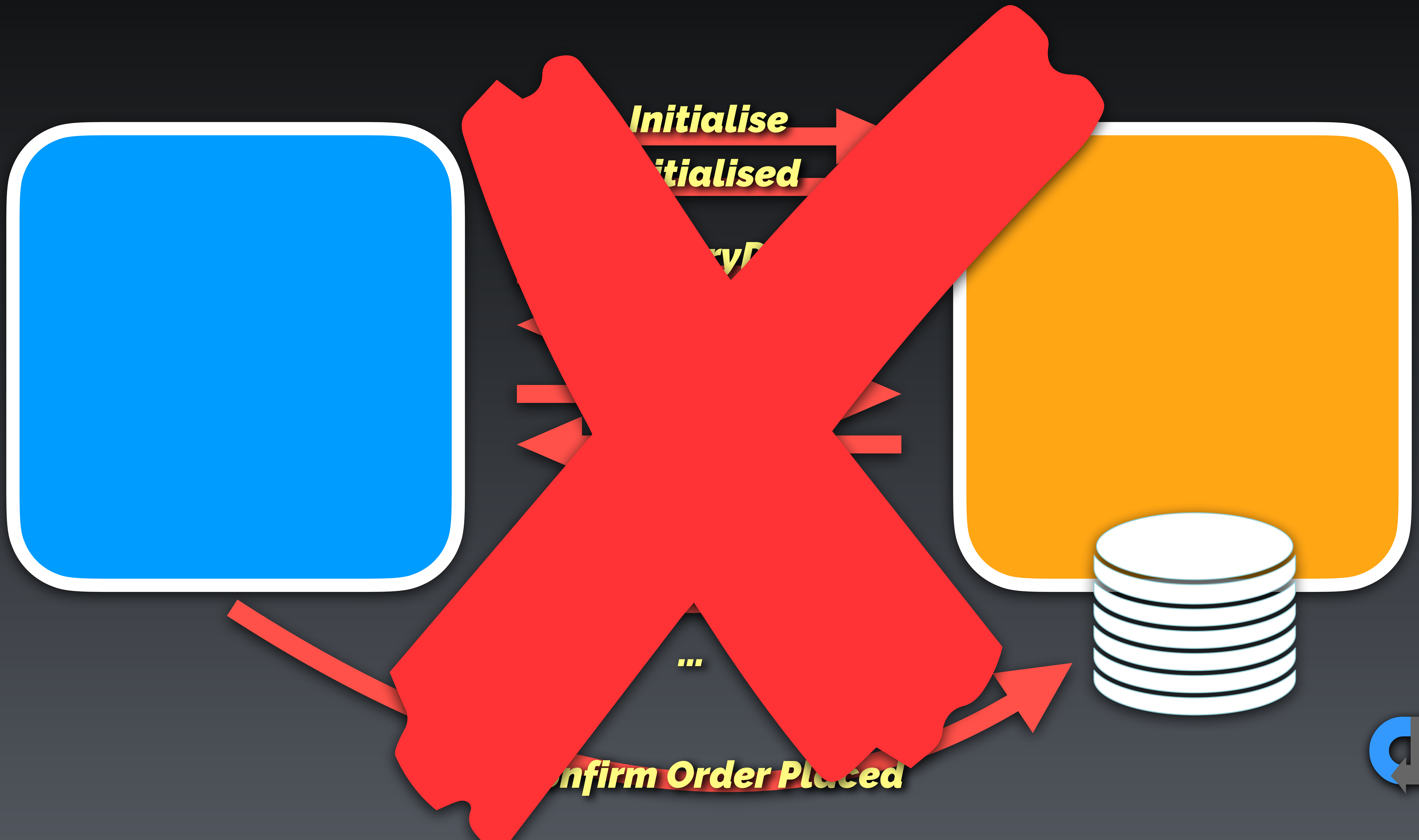


Abstraction



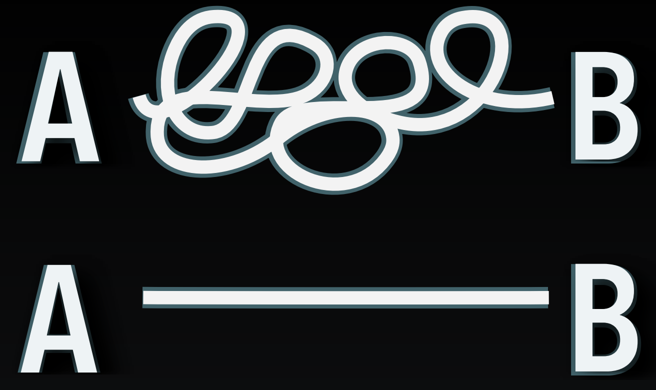
Continuous Delivery ltd

Abstraction



Continuous Delivery ltd

Abstraction

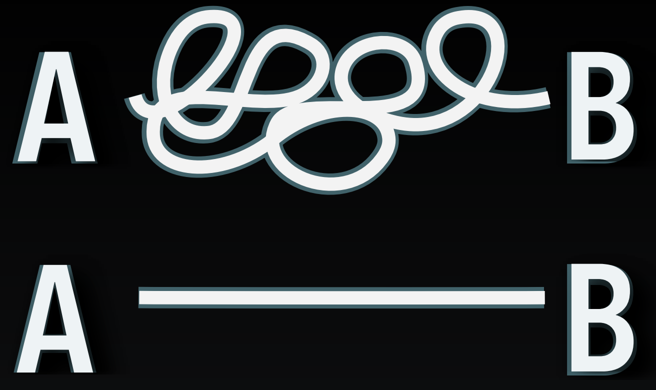


Continuous Delivery ltd

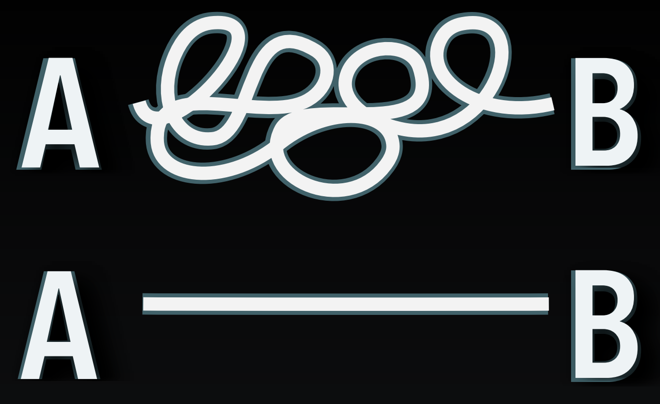
Abstraction

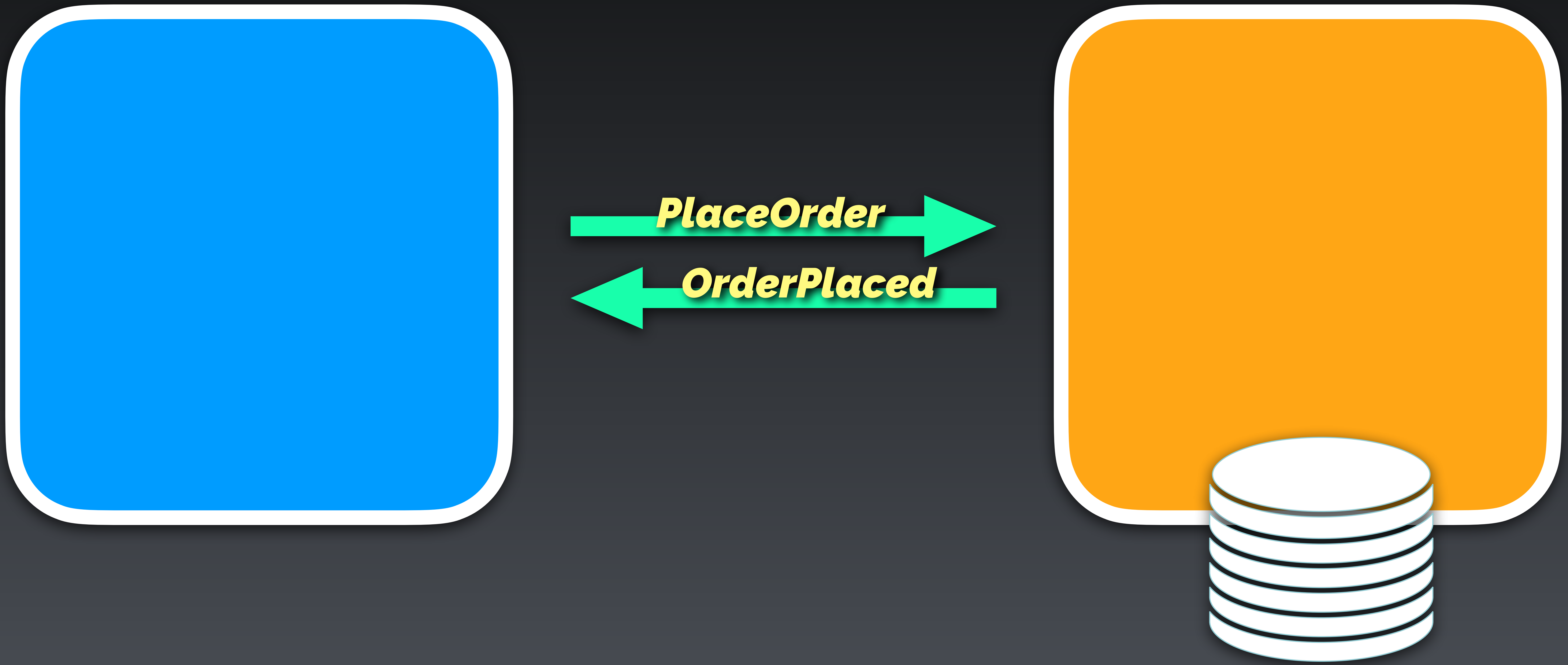
Continuous Delivery ltd

Abstraction



Abstraction





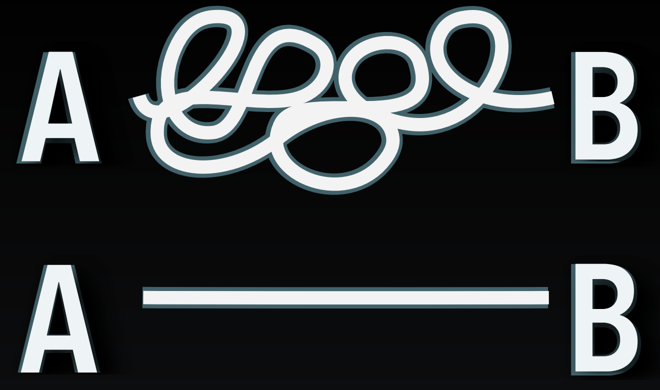
Continuous Delivery ltd

Abstraction

Keep Secrets!

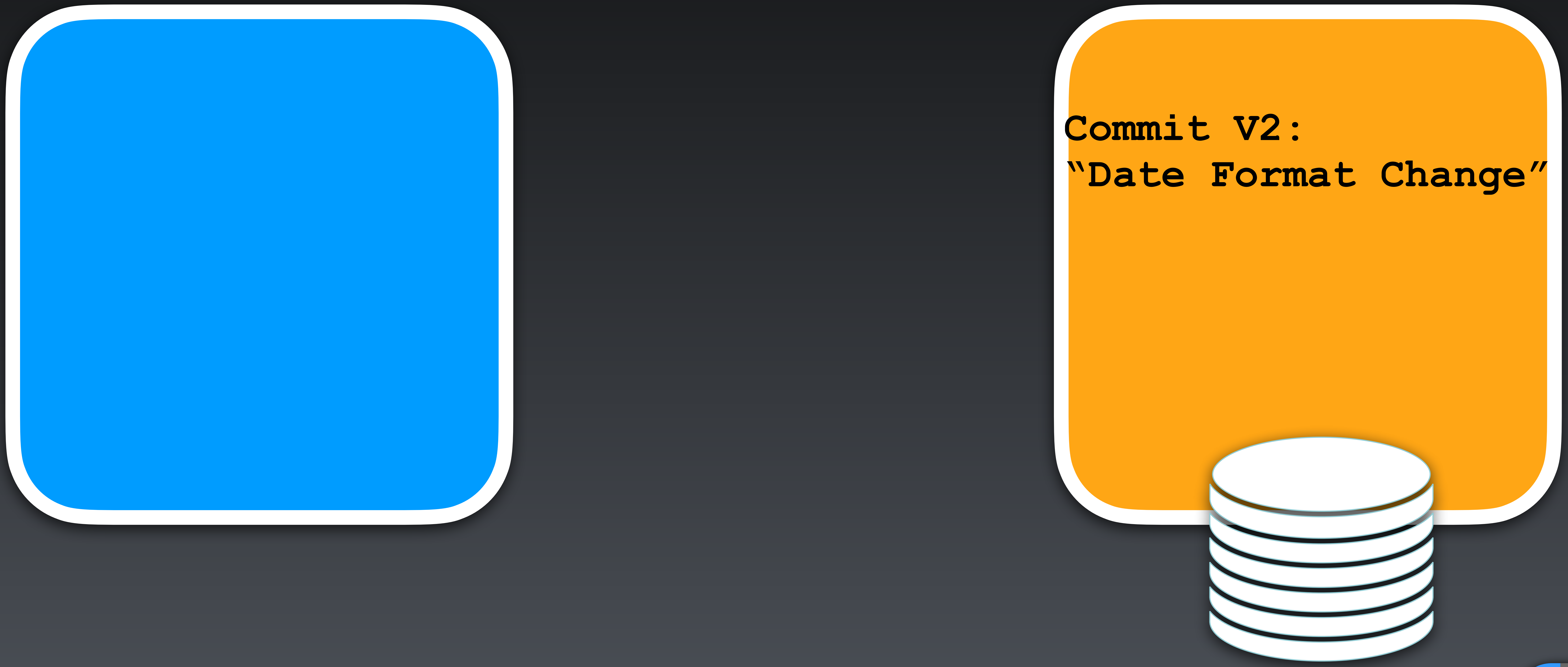
Continuous Delivery ltd

Coupling

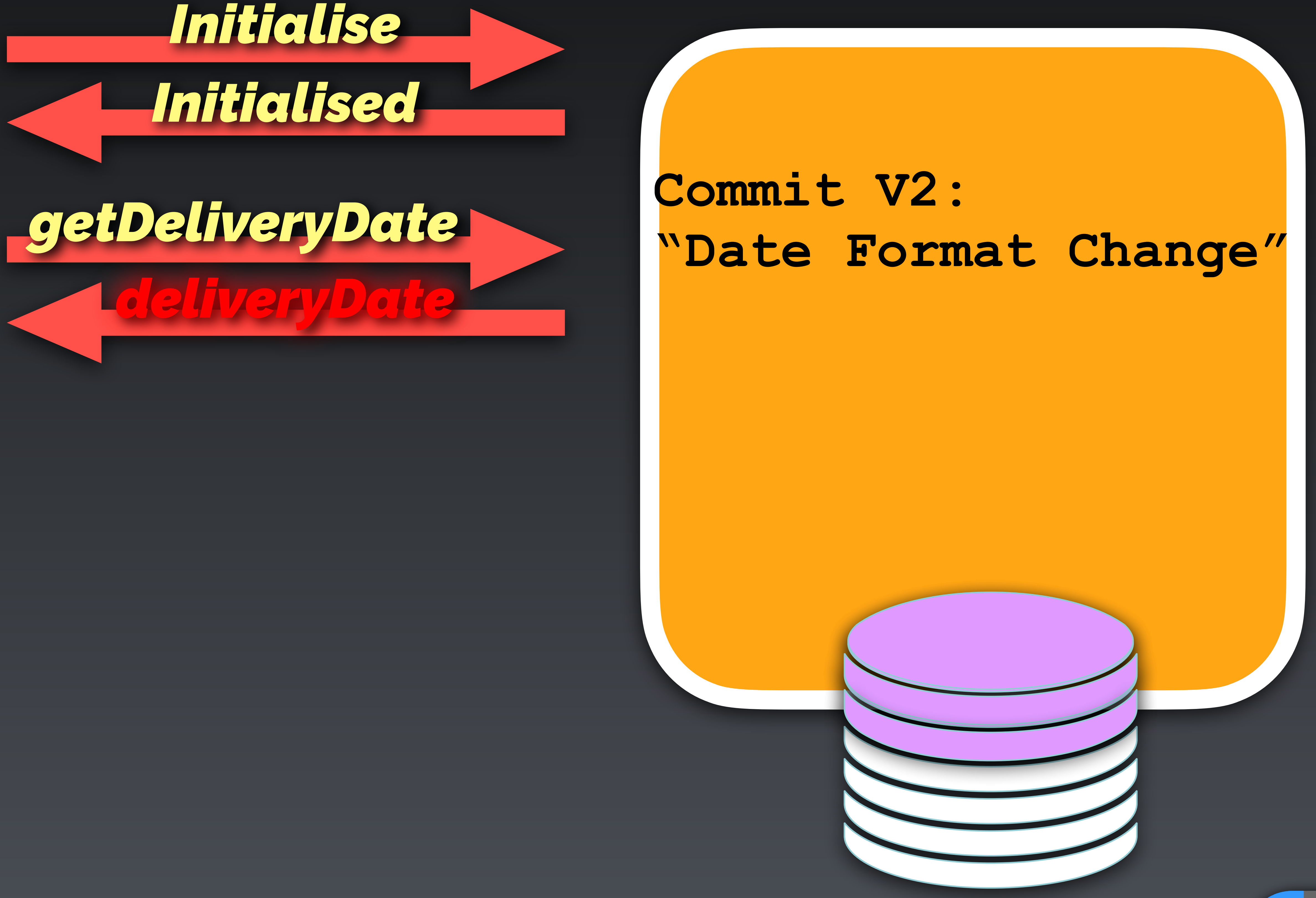


Continuous Delivery ltd

Coupling



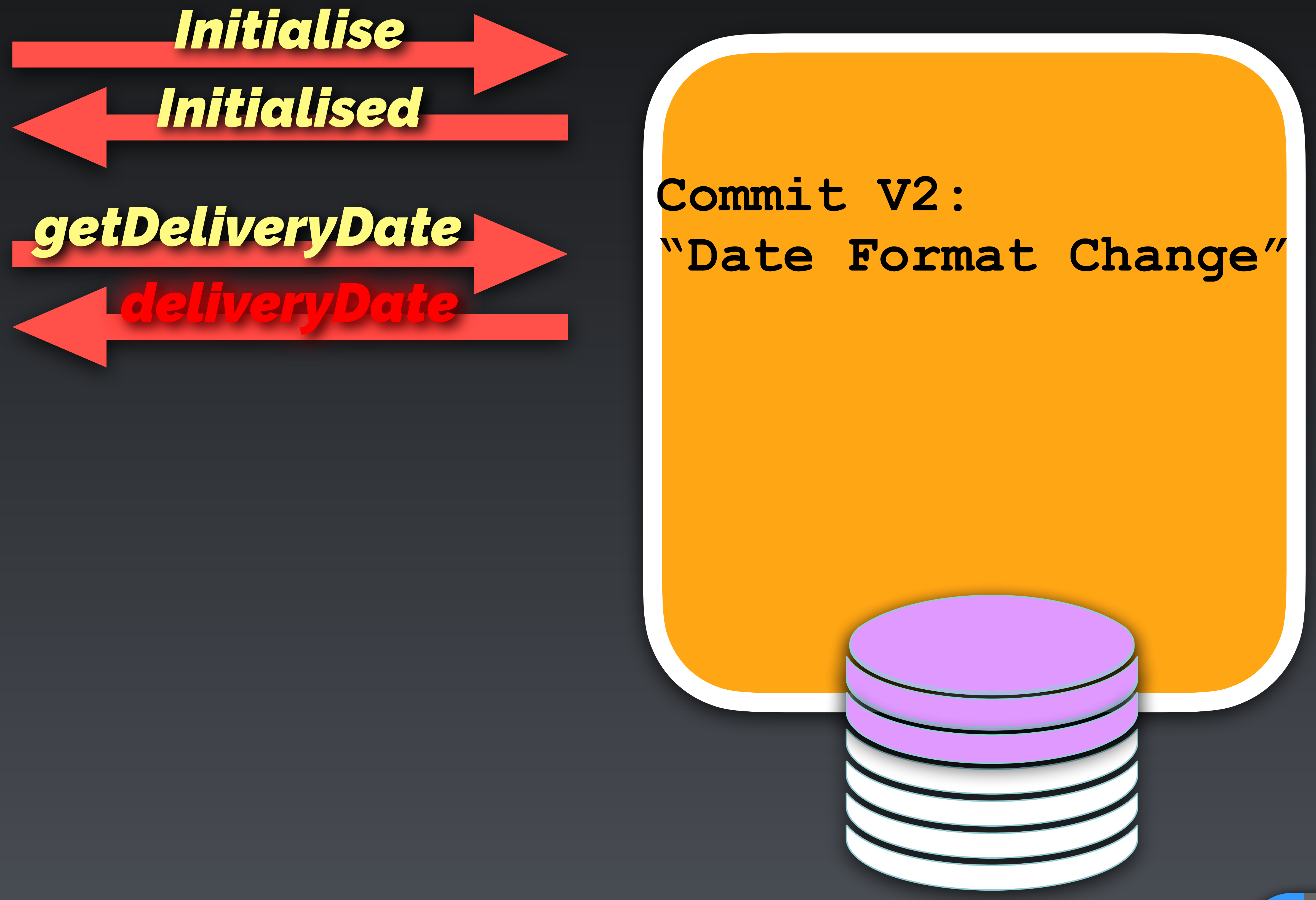
Coupling



Continuous Delivery ltd

Coupling

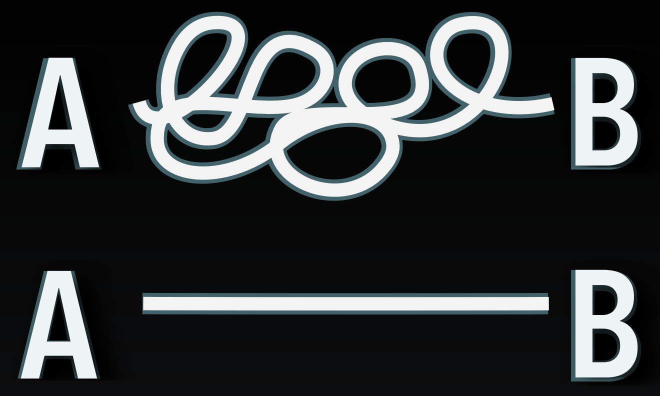
A
B
A
B


Initialise
Initialised
getDeliveryDate
deliveryDate
Commit V2:
"Date Format Change"

Coupling

Continuous Delivery ltd

Coupling

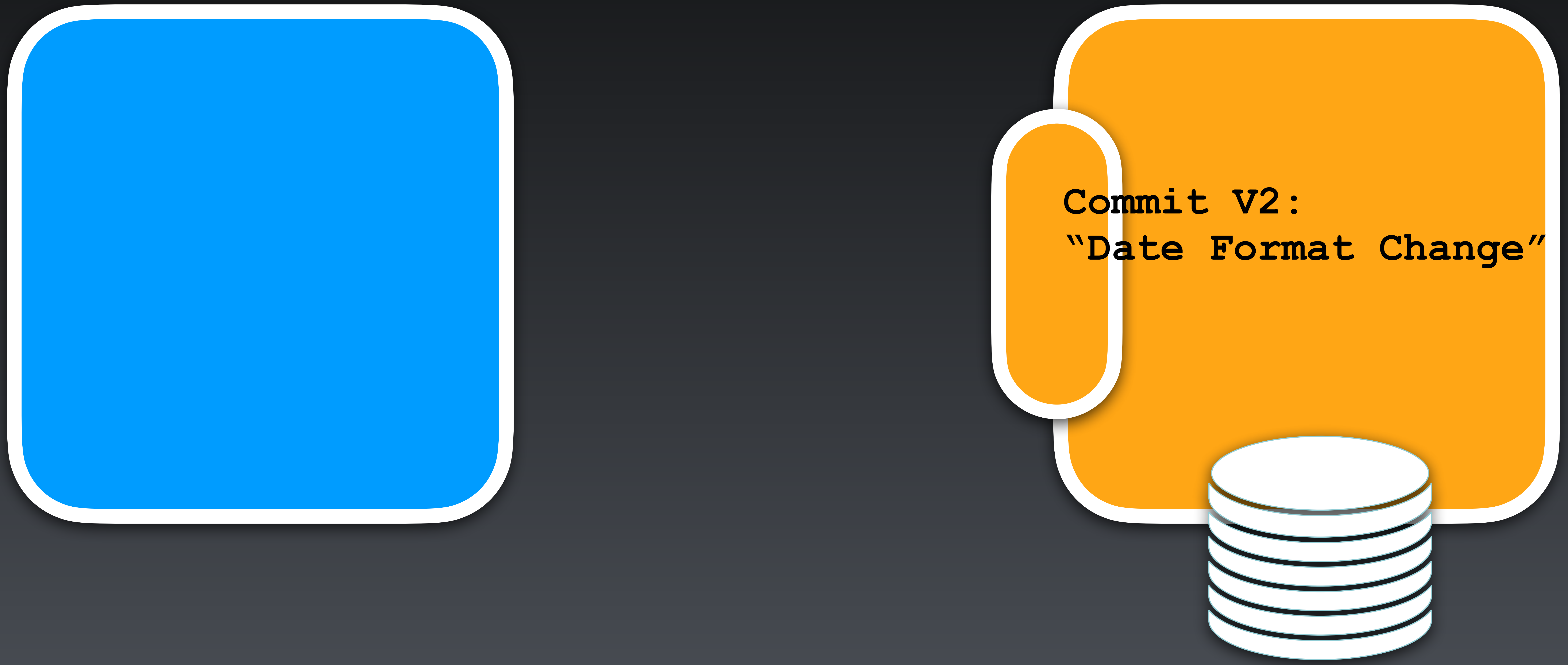


Continuous Delivery ltd

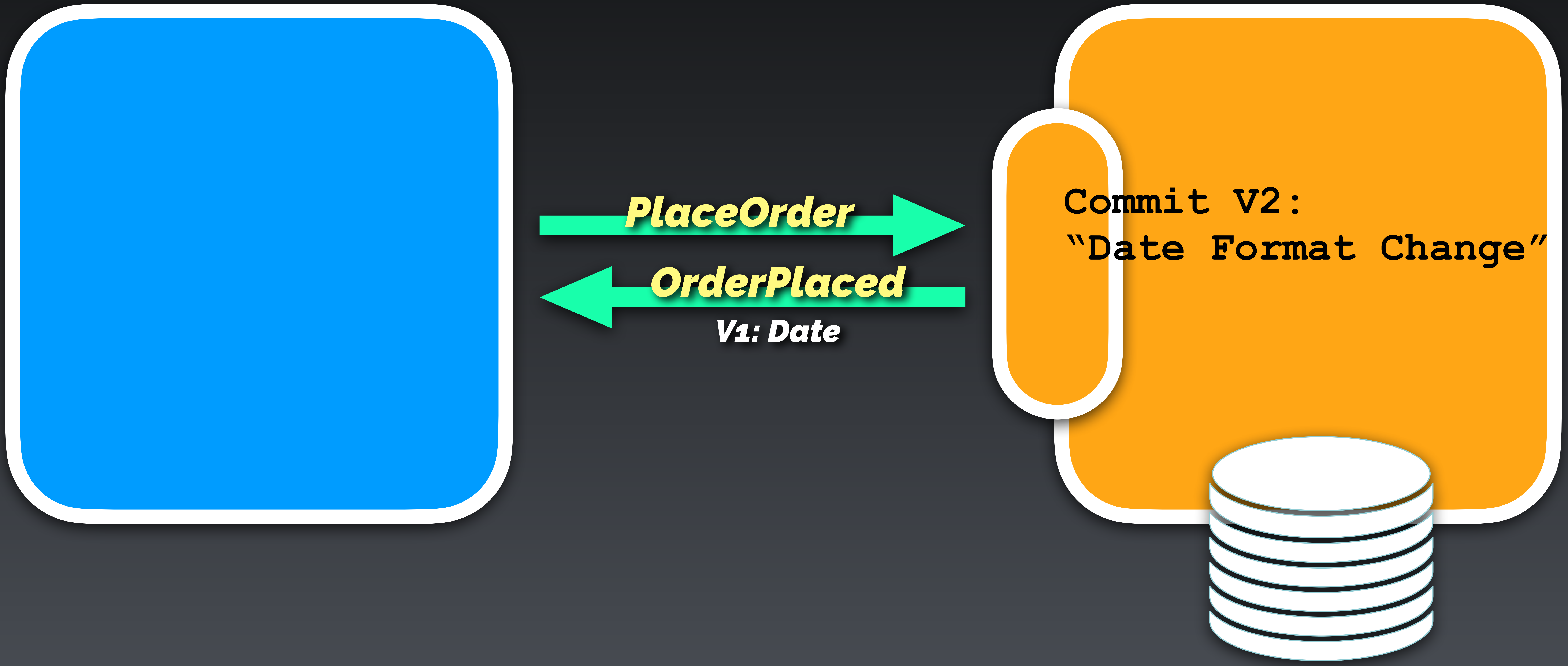
Coupling

Continuous Delivery ltd

Coupling



Coupling



Continuous Delivery ltd

Applying the Guidelines

Continuous Delivery ltd

Managing Complexity

Continuous Delivery ltd

Managing Complexity

Modularity

Cohesion

Separation of Concerns

Abstraction

Coupling

Managing Complexity

Modularity

Cohesion

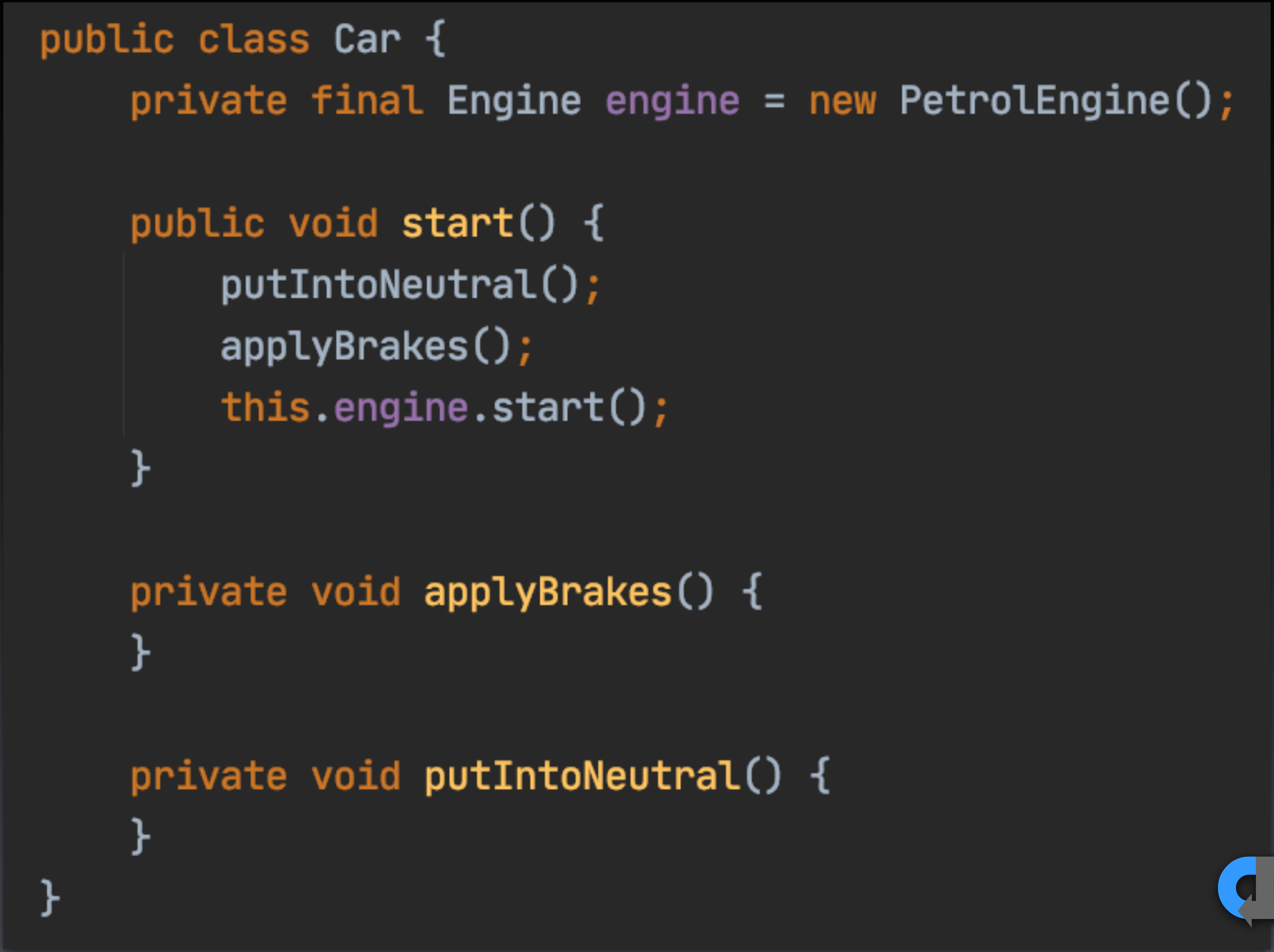
Separation of Concerns

Abstraction

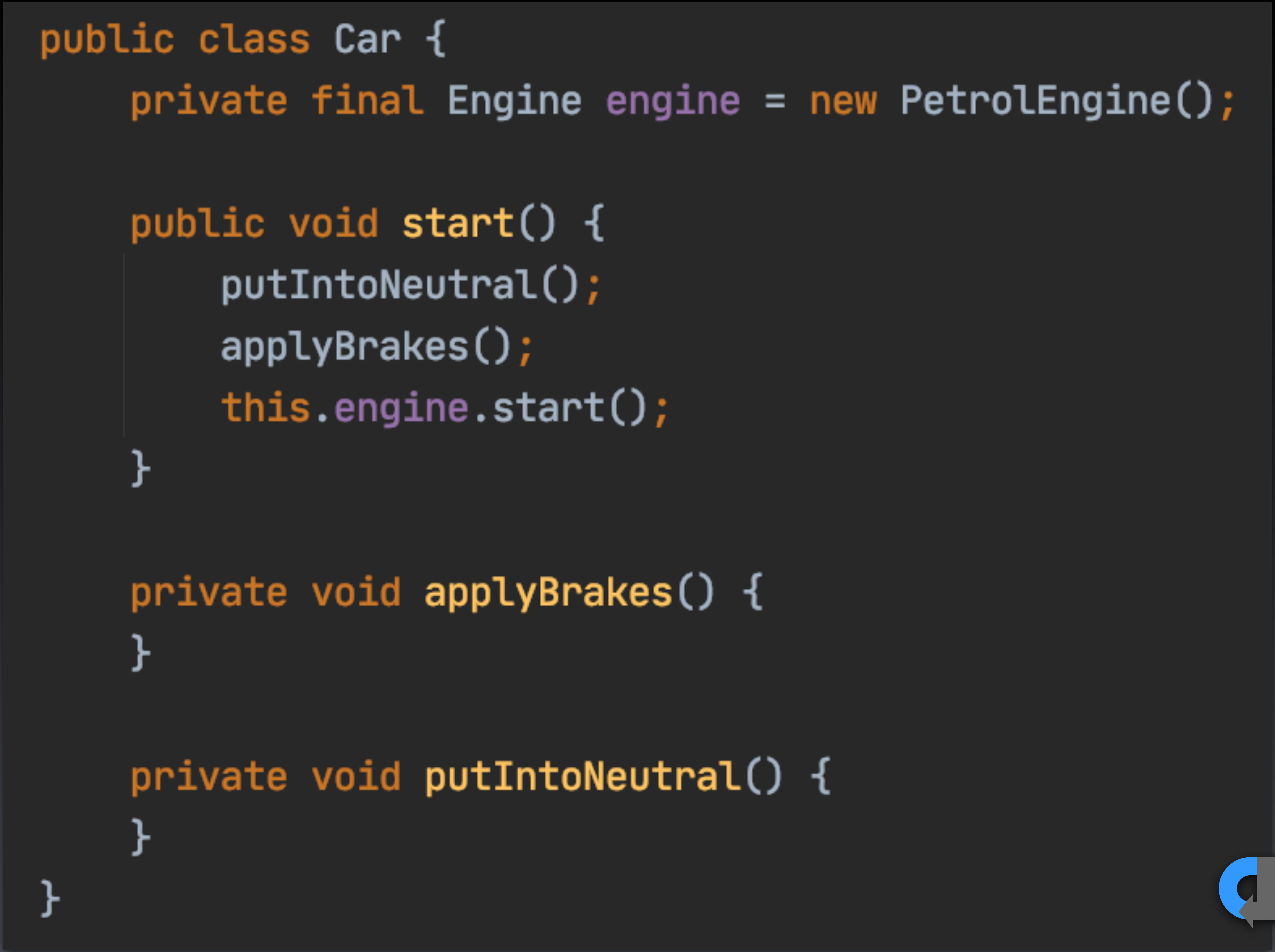
Coupling

Just Motherhood & Apple Pie?

Continuous Delivery ltd

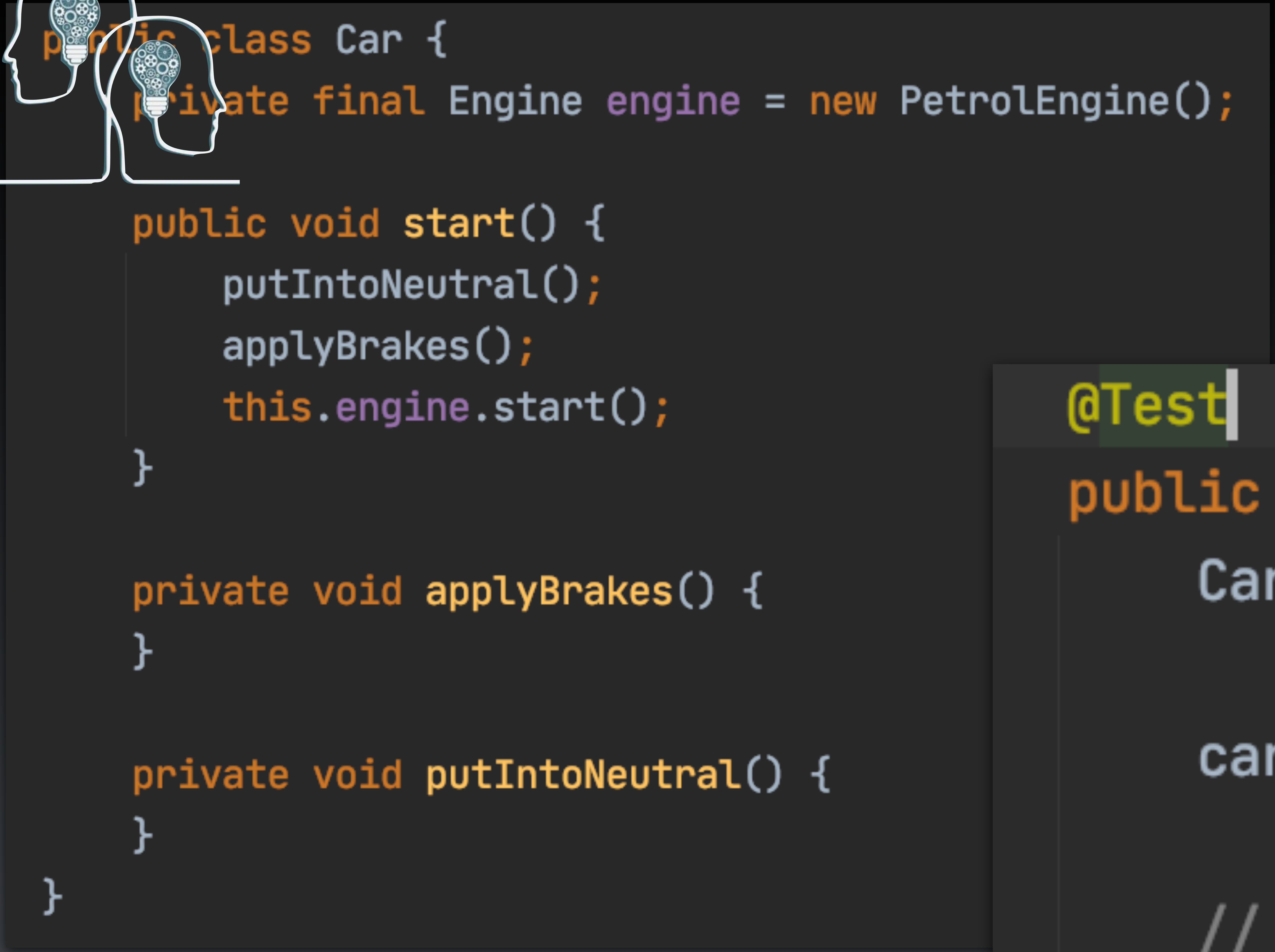


```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

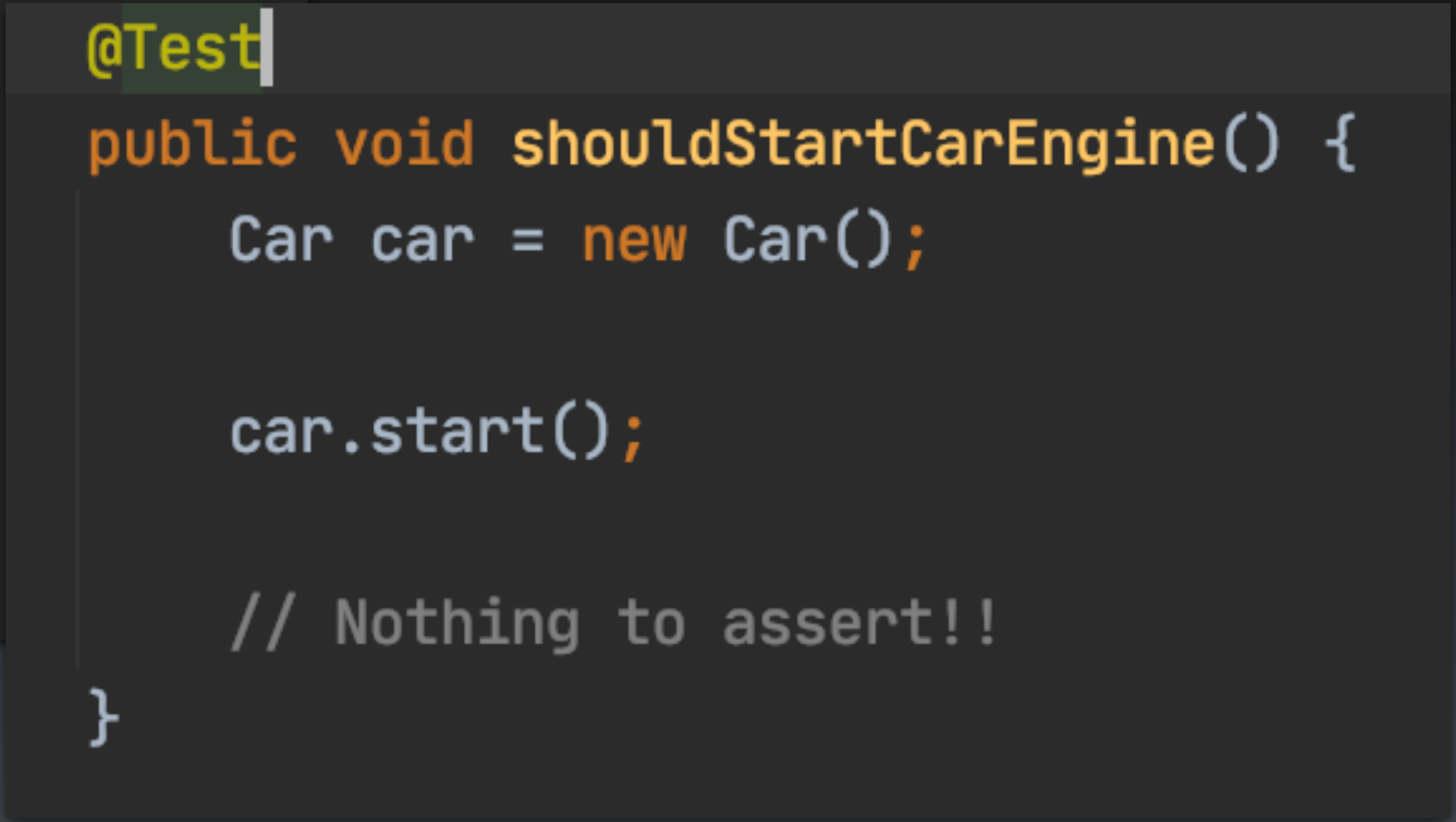


```java
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

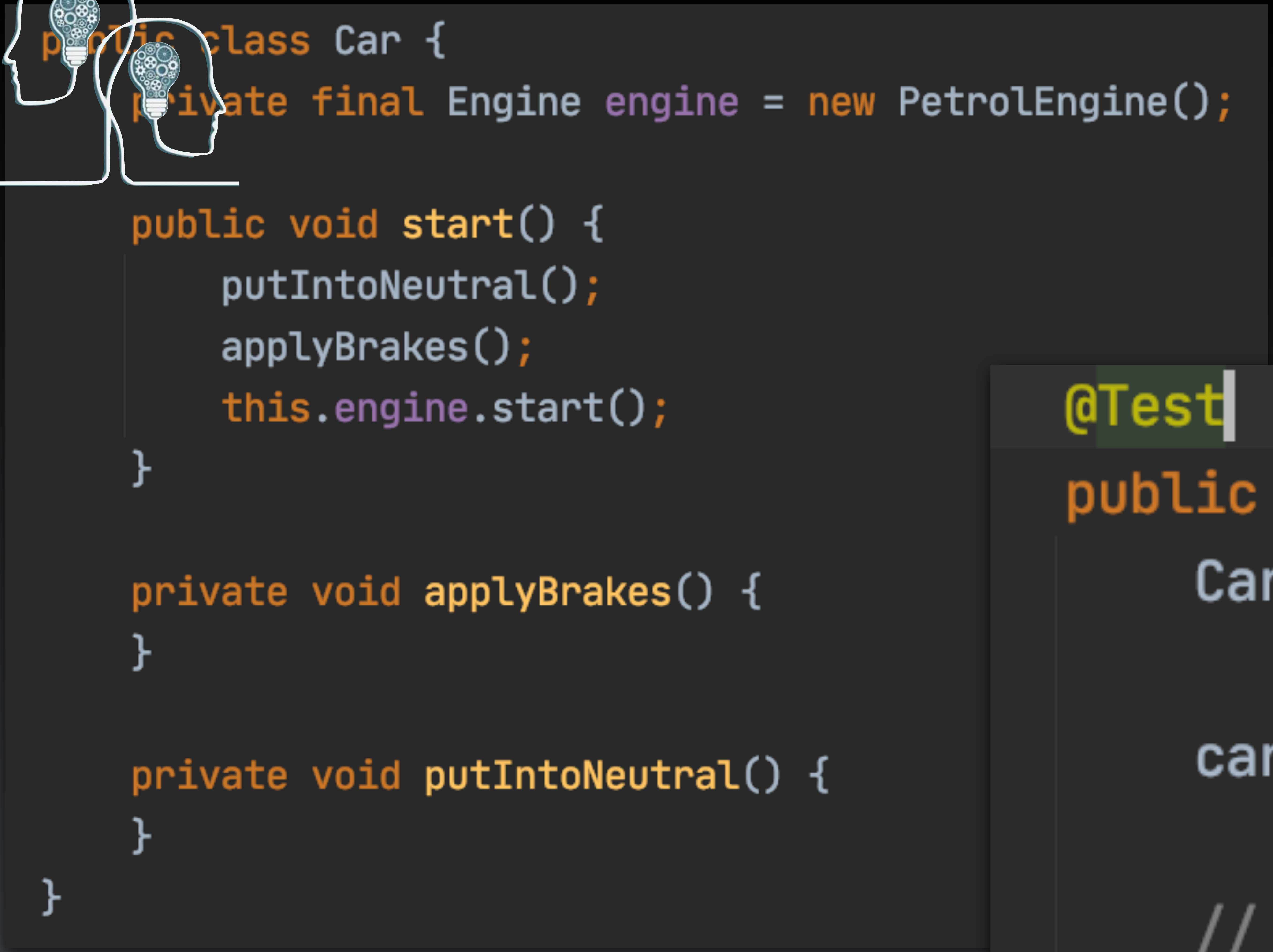
Continuous Delivery ltd



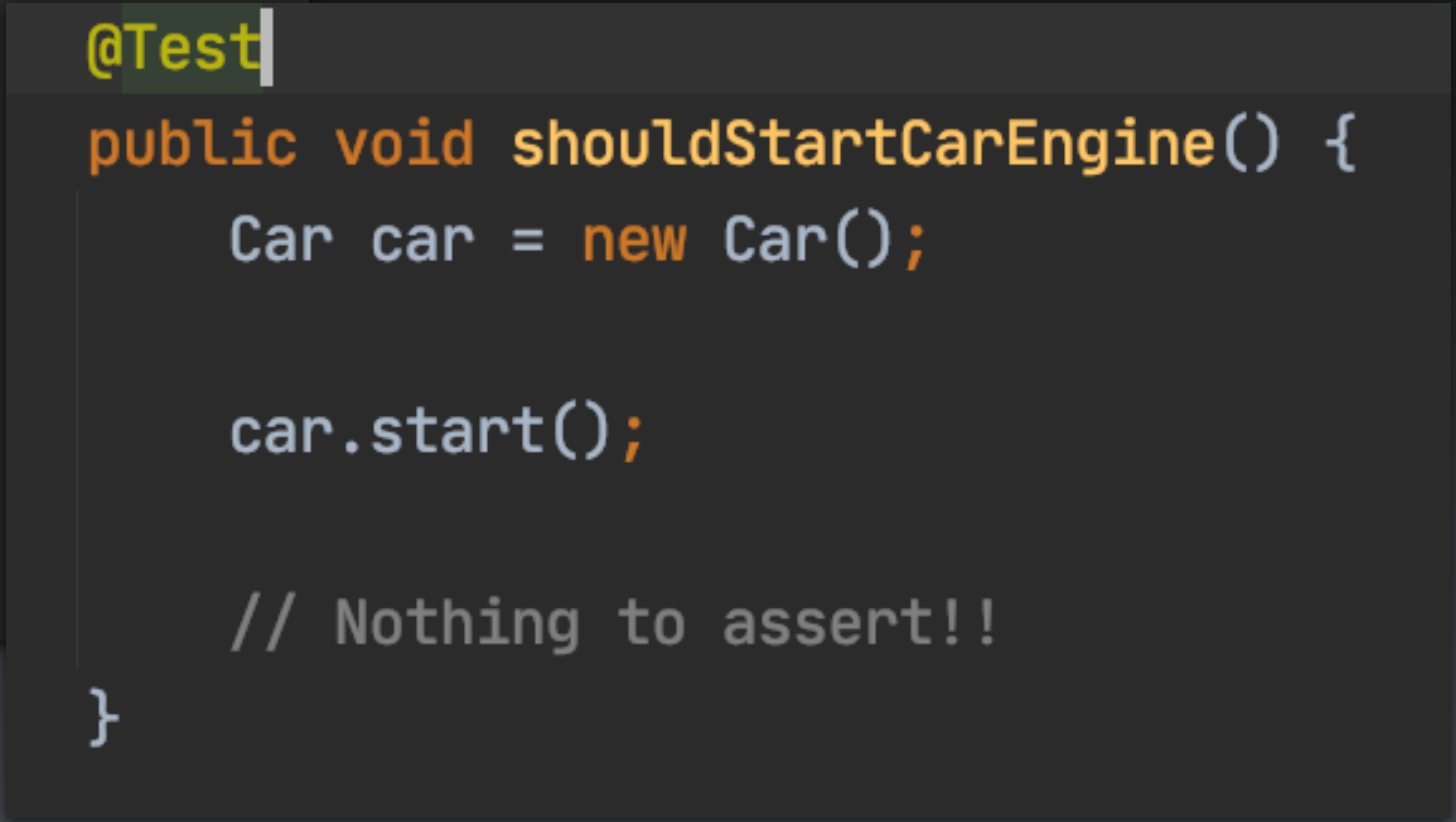
```
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

    // Nothing to assert!!
}
```

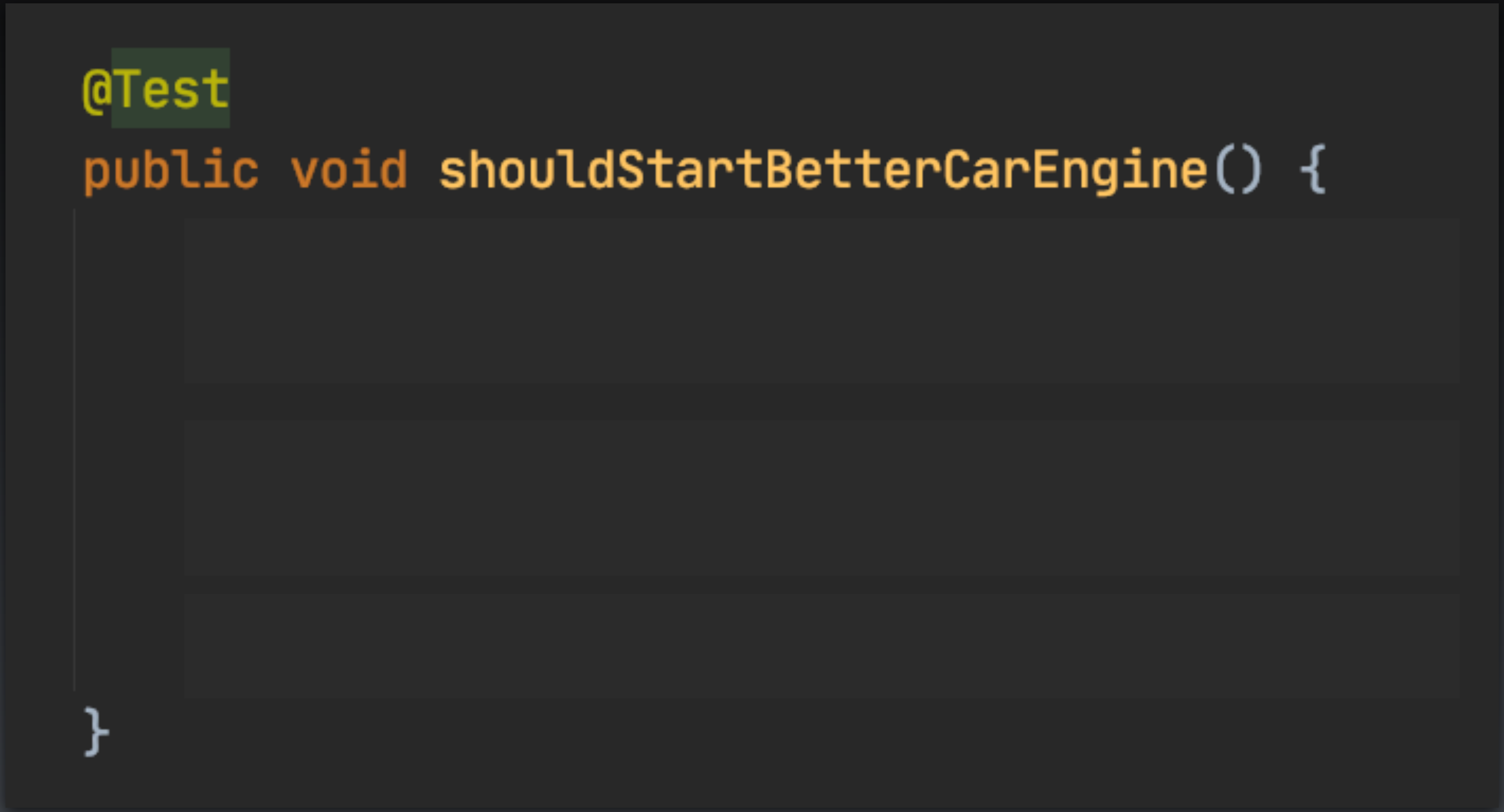


```
public class Car {
    private final Engine engine = new PetrolEngine();

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

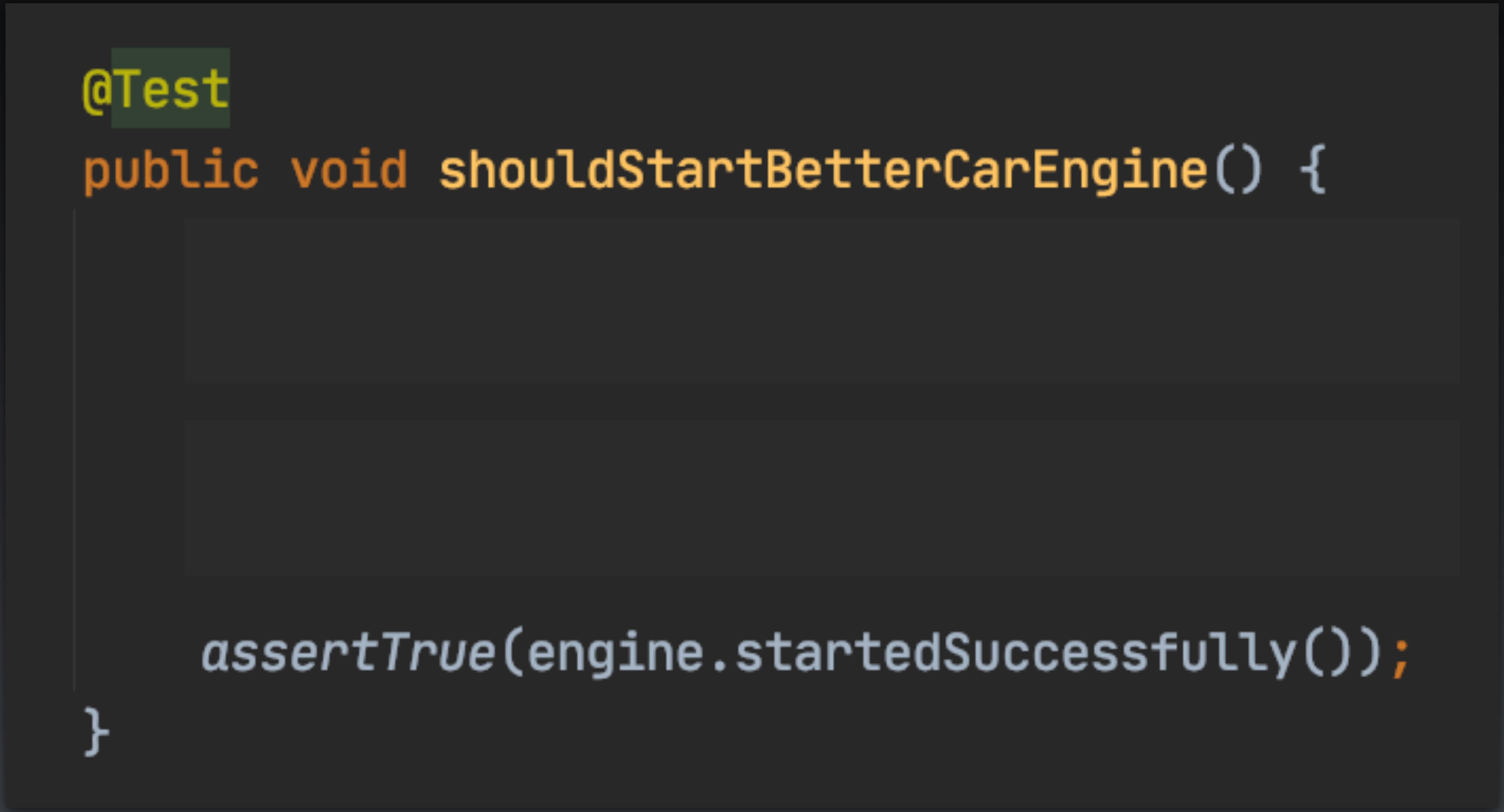


```
@Test
public void shouldStartCarEngine() {
    Car car = new Car();

    car.start();

    // Nothing to assert!!
}
```

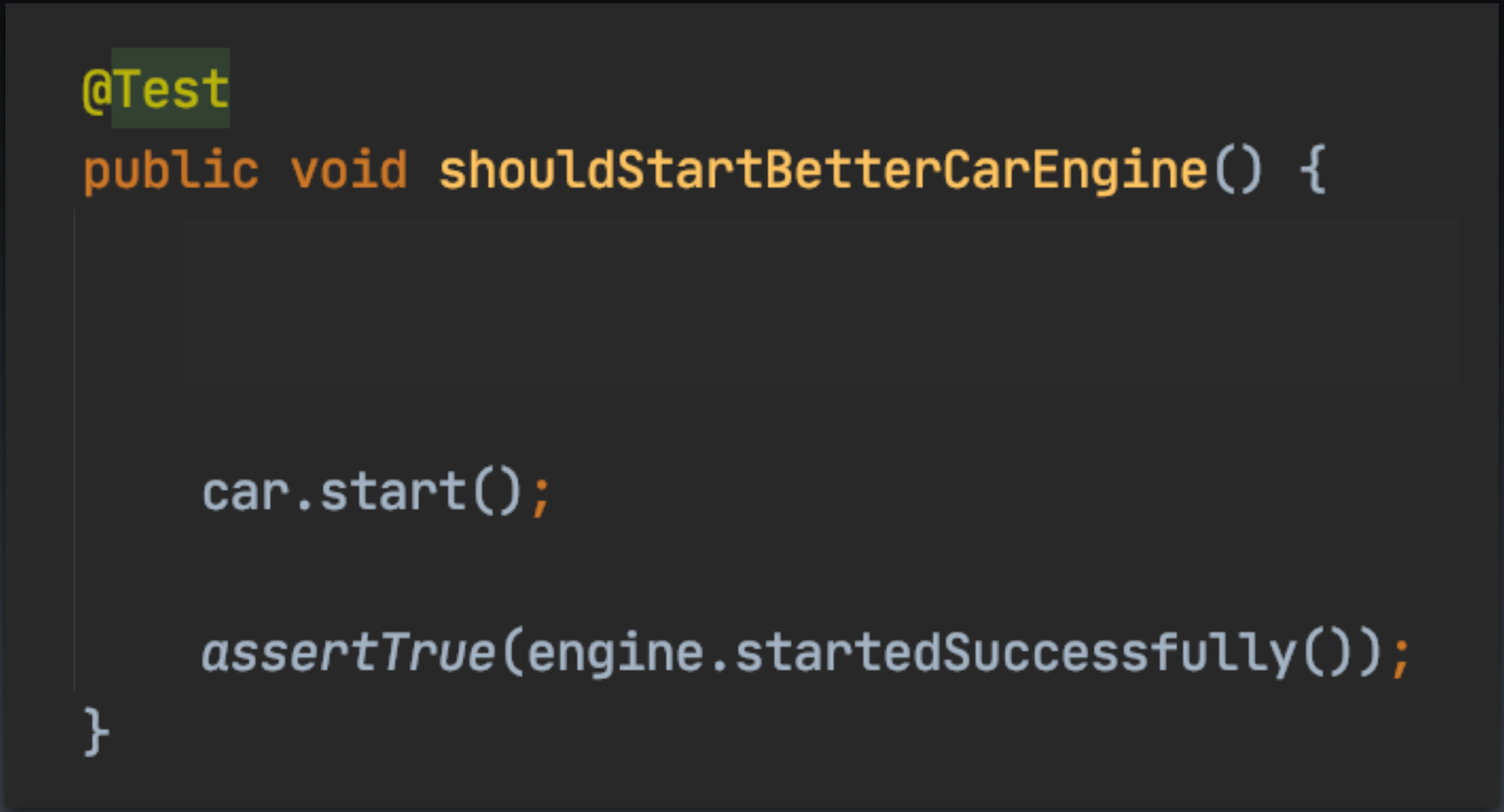
Continuous Delivery ltd



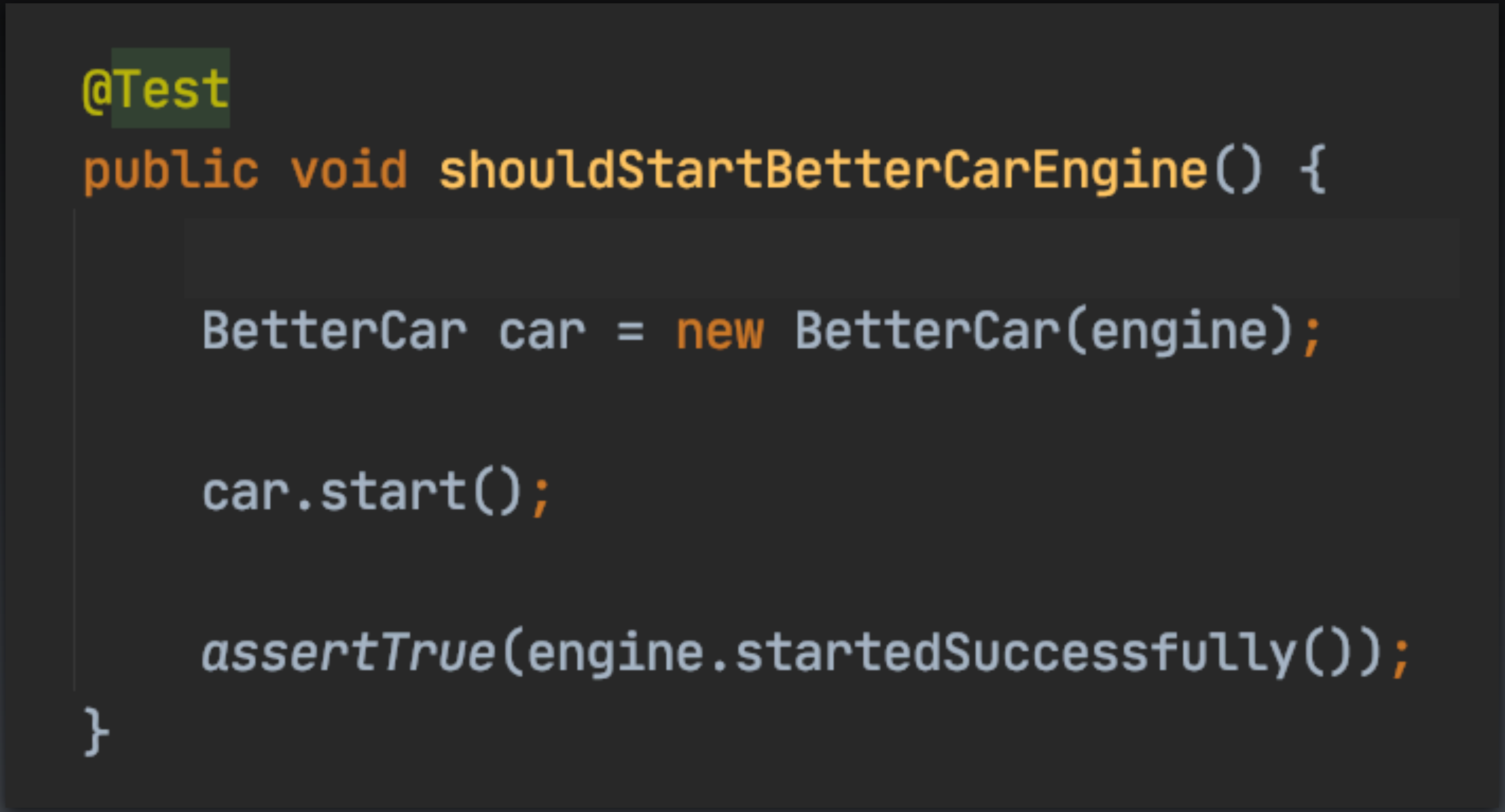
```
@Test
public void shouldStartBetterCarEngine() {



}
```



```
@Test
public void shouldStartBetterCarEngine() {



    assertTrue(engine.startedSuccessfully());
}
```

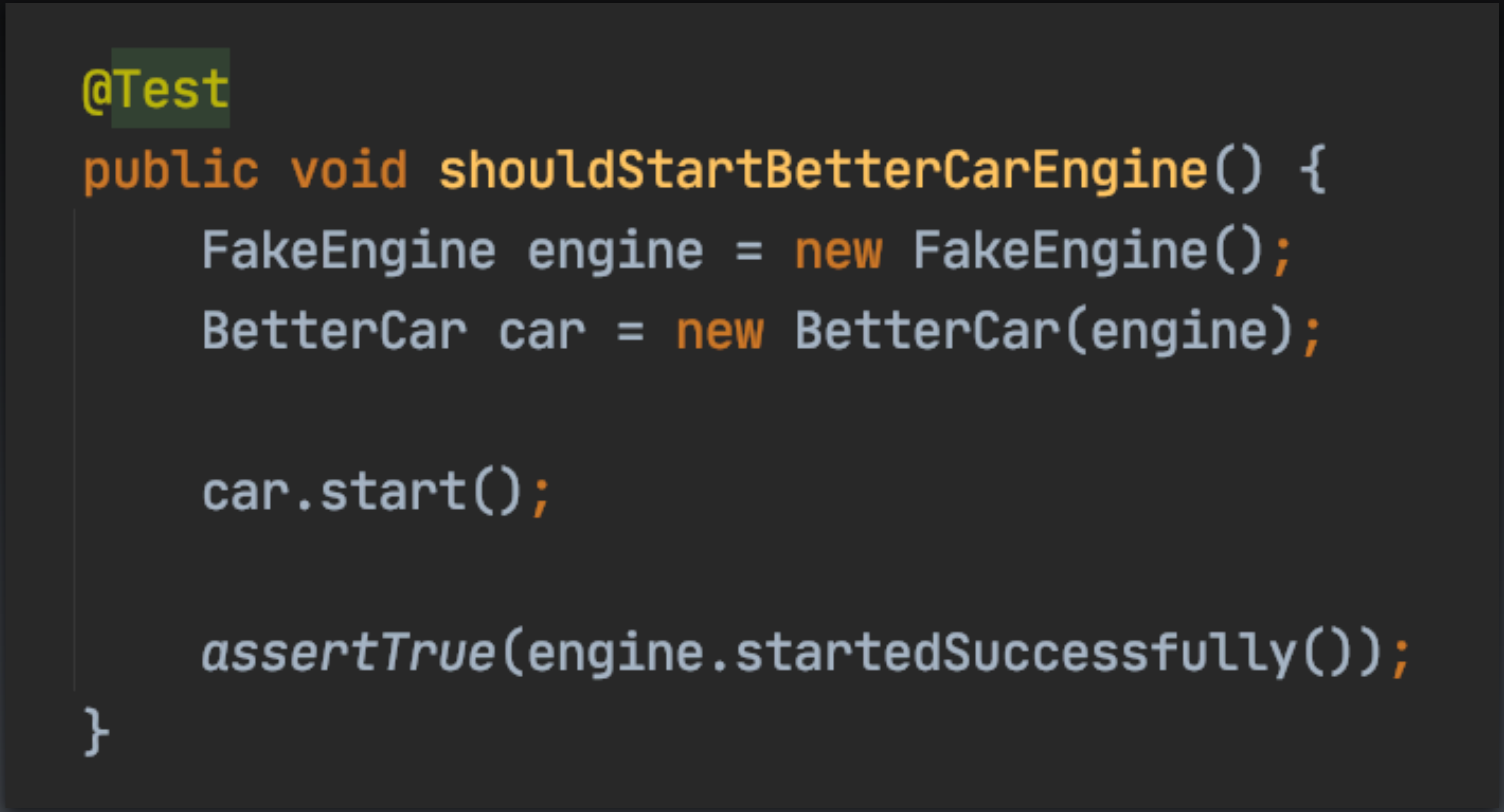


```
@Test
public void shouldStartBetterCarEngine() {



    car.start();

    assertTrue(engine.startedSuccessfully());
}
```



```java
@Test
public void shouldStartBetterCarEngine() {

    BetterCar car = new BetterCar(engine);

    car.start();

    assertTrue(engine.startedSuccessfully());
}
```

Continuous Delivery ltd



```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

    assertTrue(engine.startedSuccessfully());
}
```

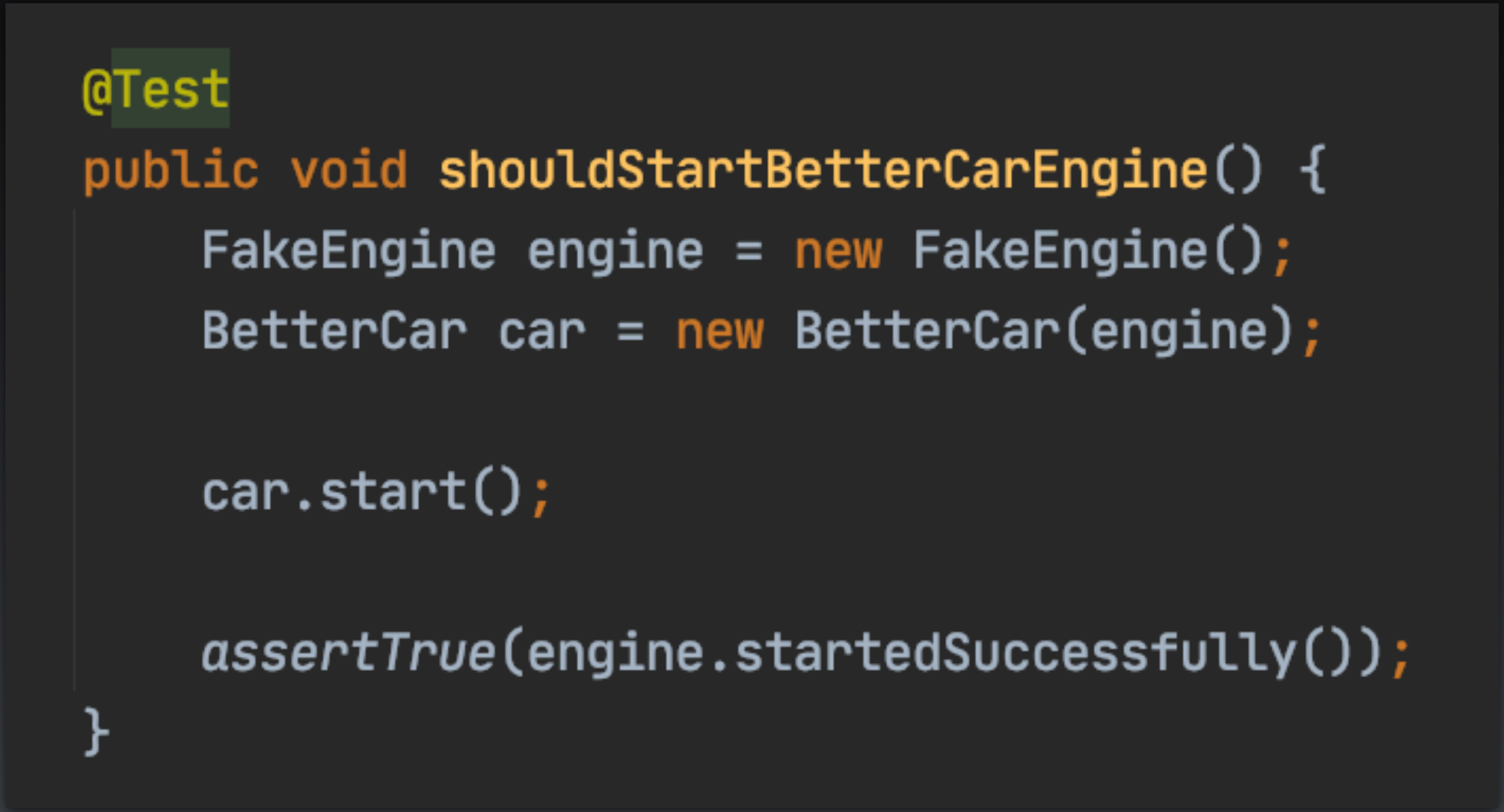
A ~ B

A — B

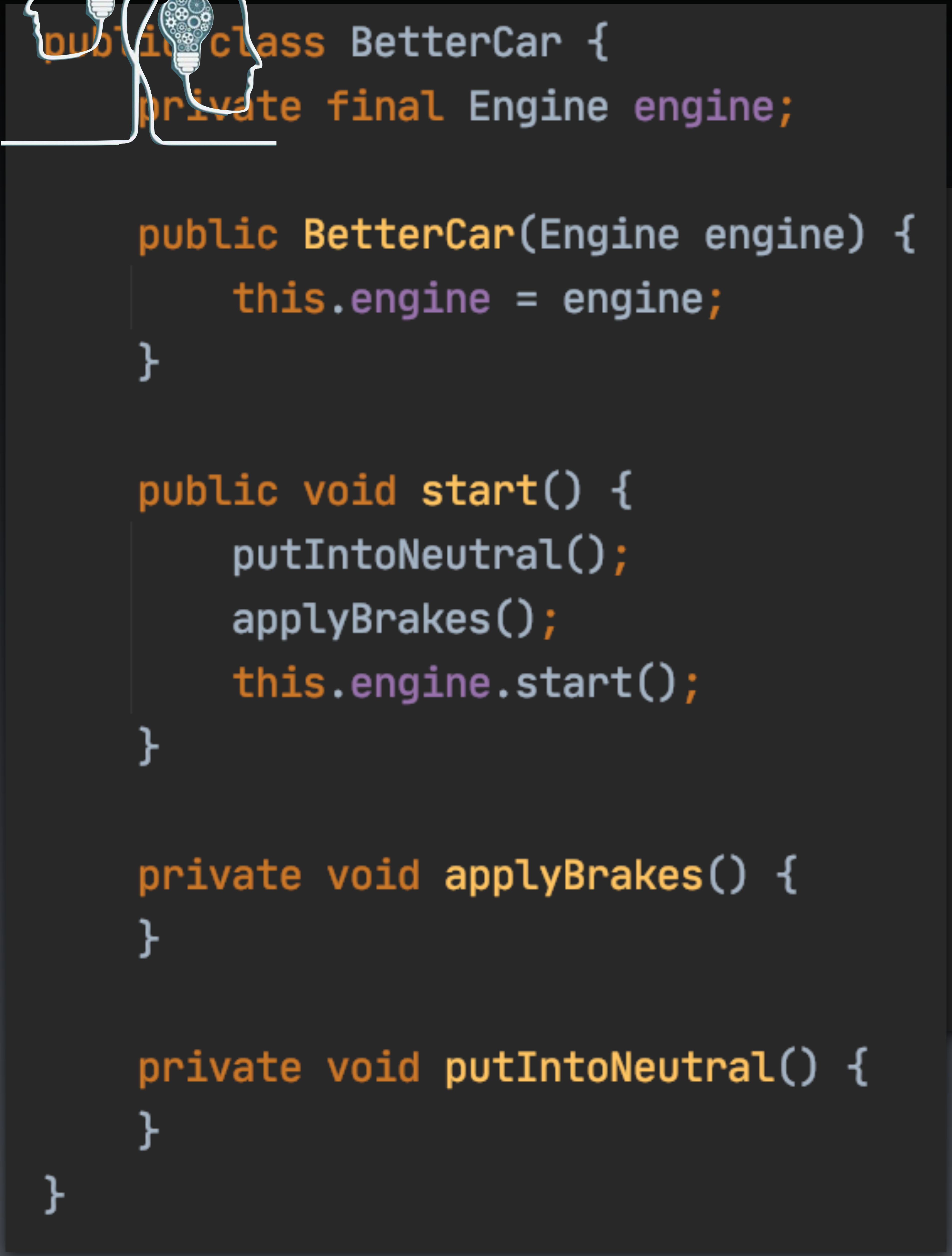
```java
public class FakeEngine implements Engine {
    private boolean started = false;

    @Override
    public void start() {
        started = true;
    }

    public boolean startedSuccessfully() {
        return started;
    }
}
```

```
{
;
);

y());
```

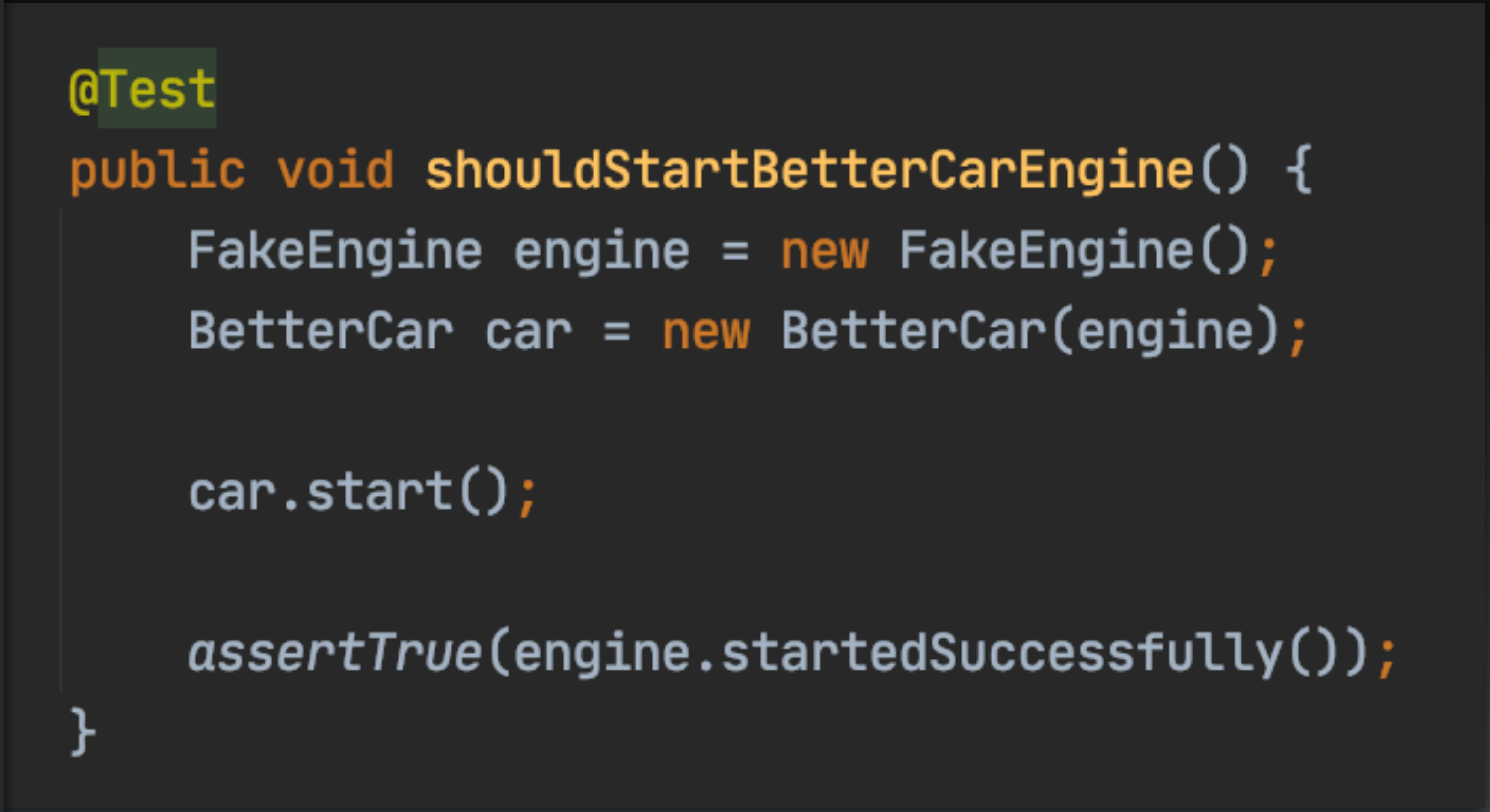
Continuous Delivery ltd



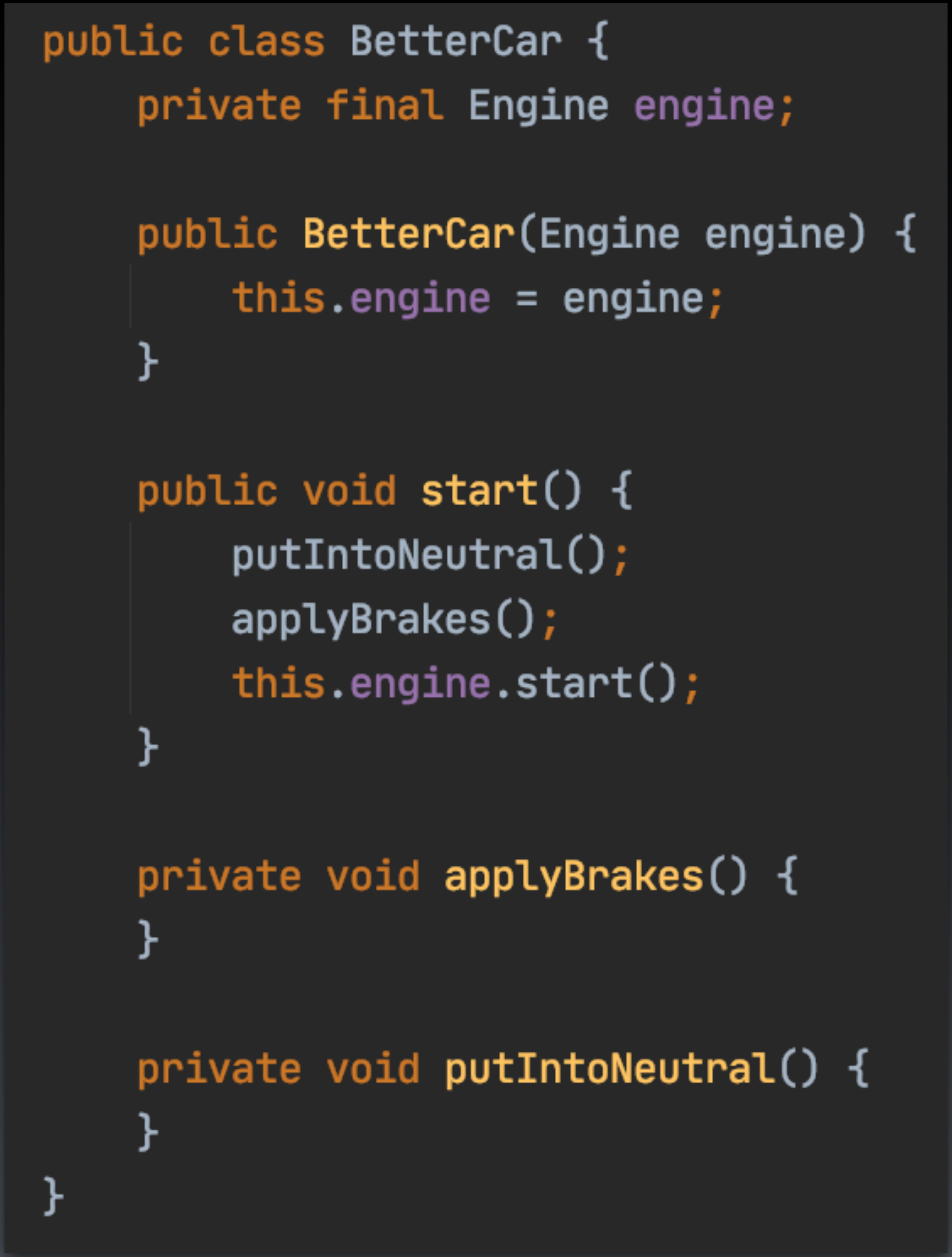
```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

    assertTrue(engine.startedSuccessfully());
}
```



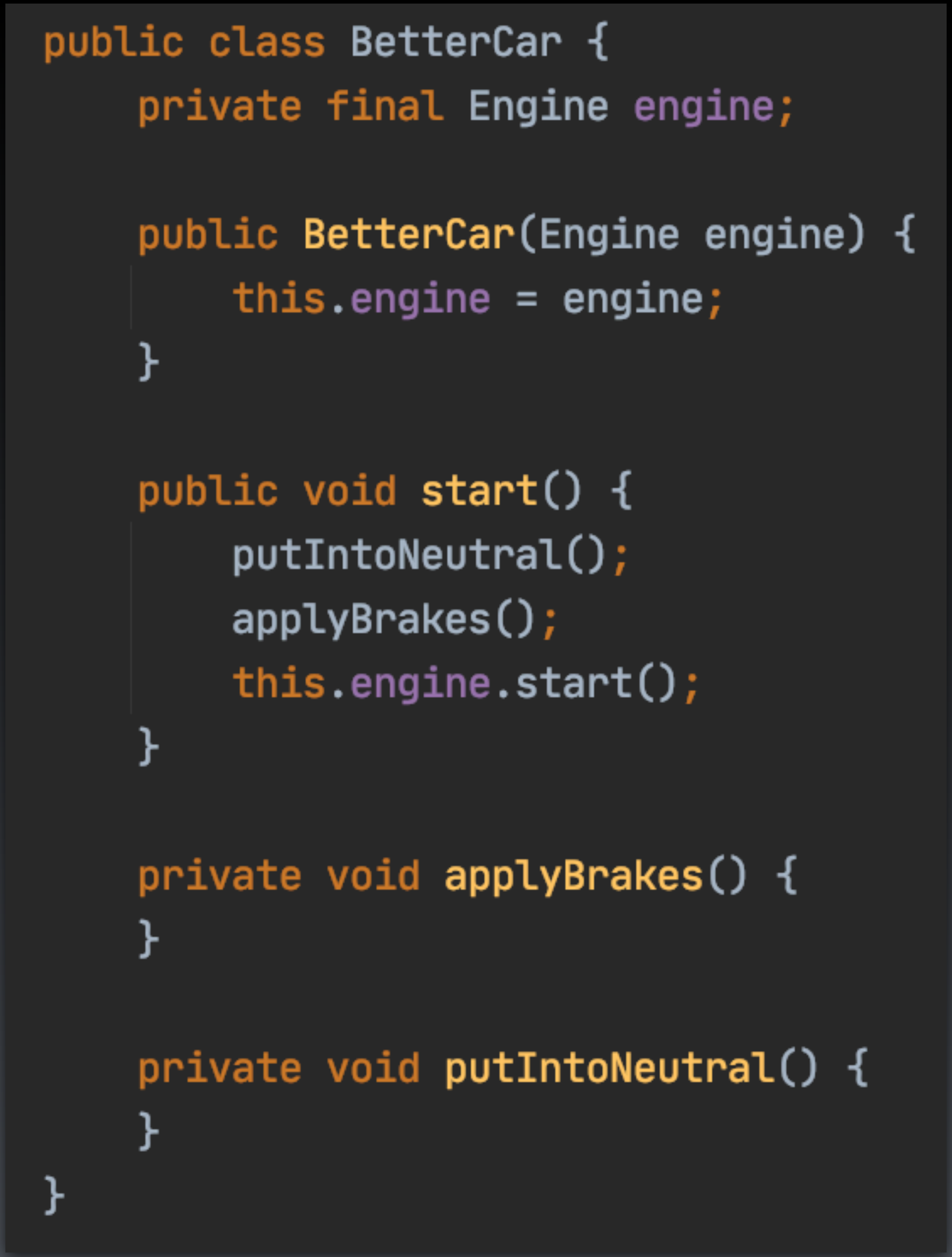
```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```



```java
@Test
public void shouldStartBetterCarEngine() {
    FakeEngine engine = new FakeEngine();
    BetterCar car = new BetterCar(engine);

    car.start();

    assertTrue(engine.startedSuccessfully());
}
```

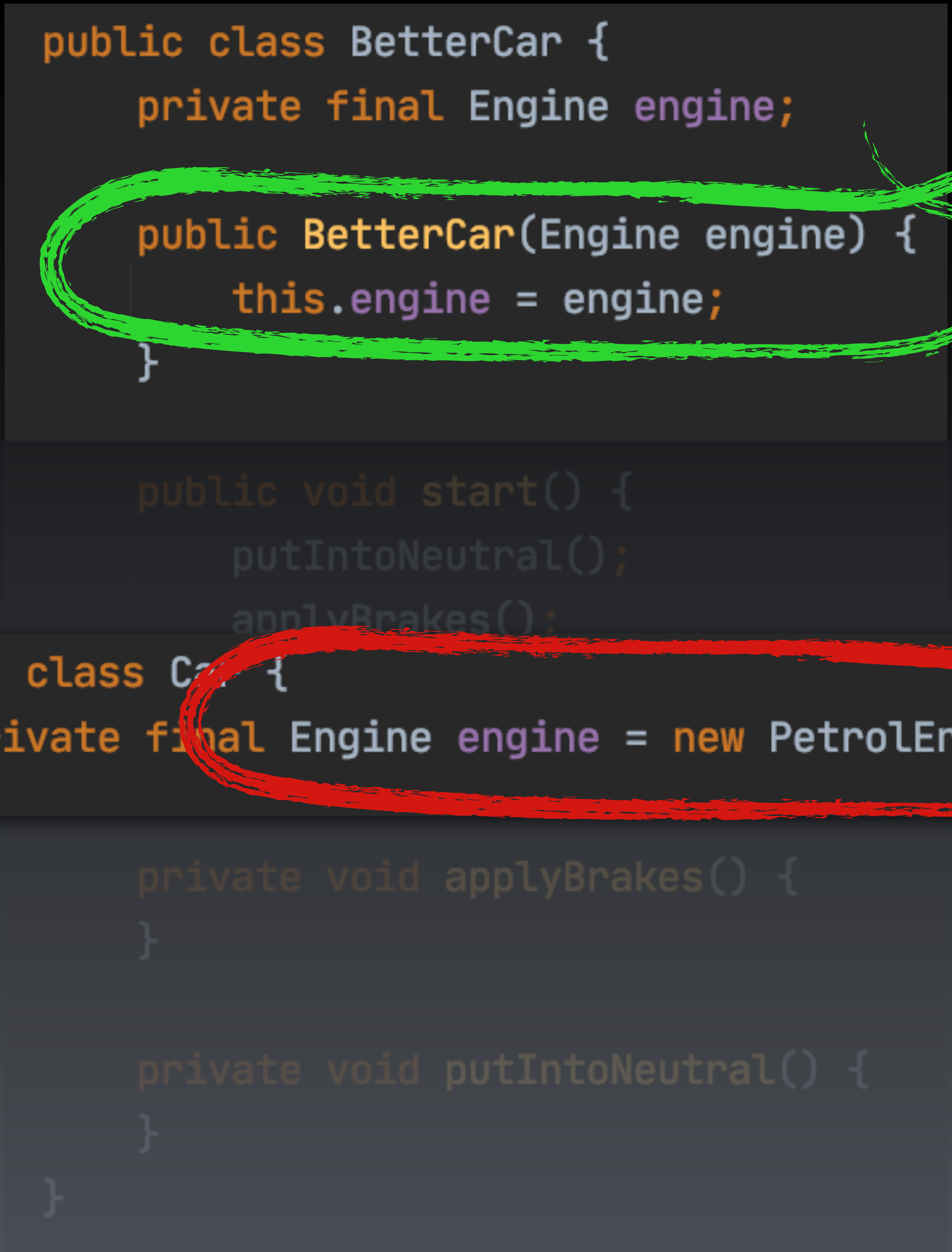


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

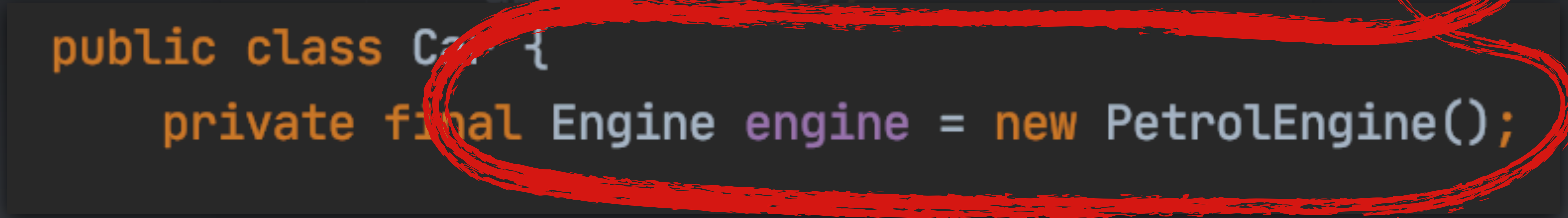


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

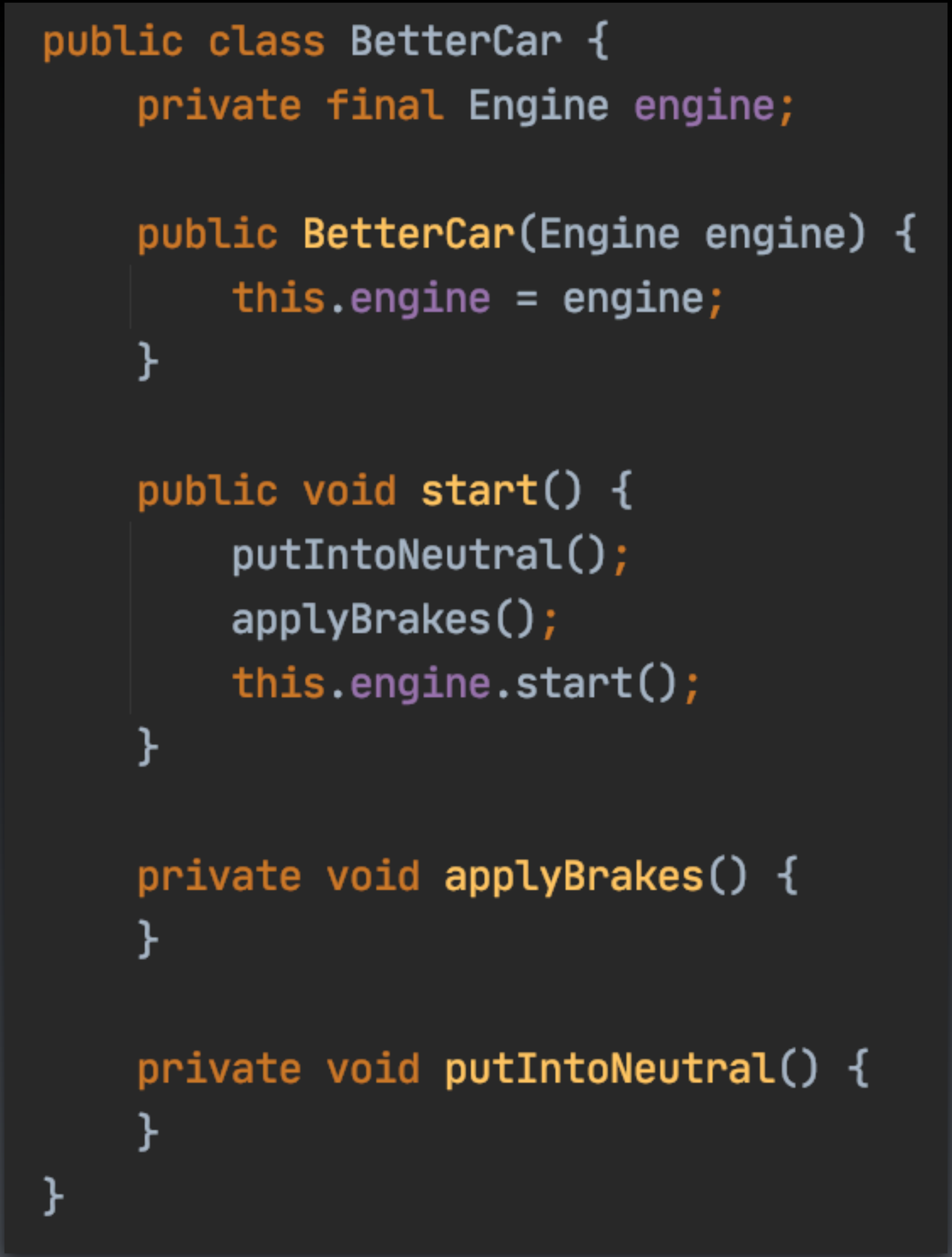
Continuous Delivery ltd



```
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

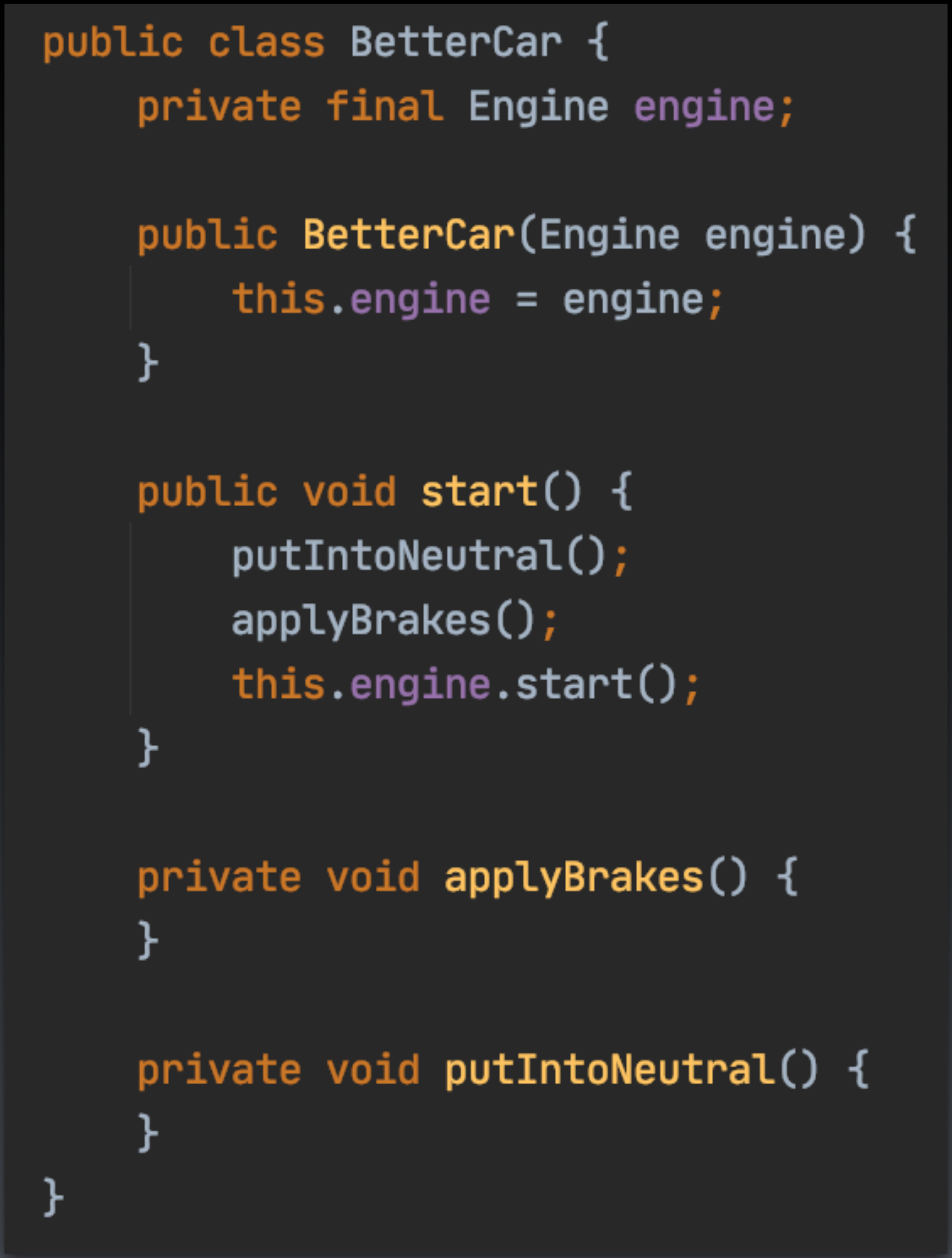


```
public class Car {
    private final Engine engine = new PetrolEngine();
```



```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

Continuous Delivery ltd



```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }

    public void start() {
        putIntoNeutral();
        applyBrakes();
        this.engine.start();
    }

    private void applyBrakes() {
    }

    private void putIntoNeutral() {
    }
}
```

A B

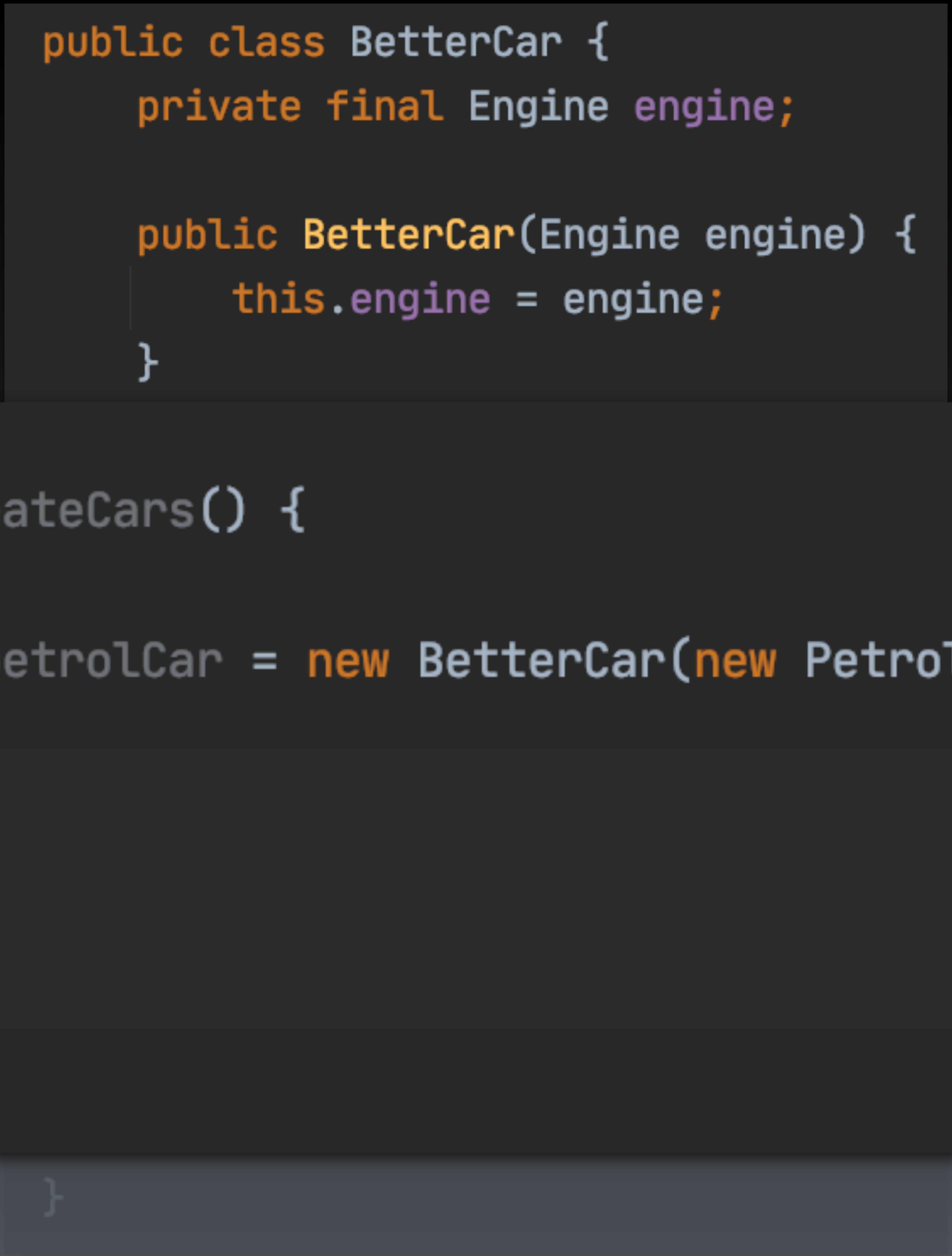
A B

Continuous Delivery ltd

```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
}
```

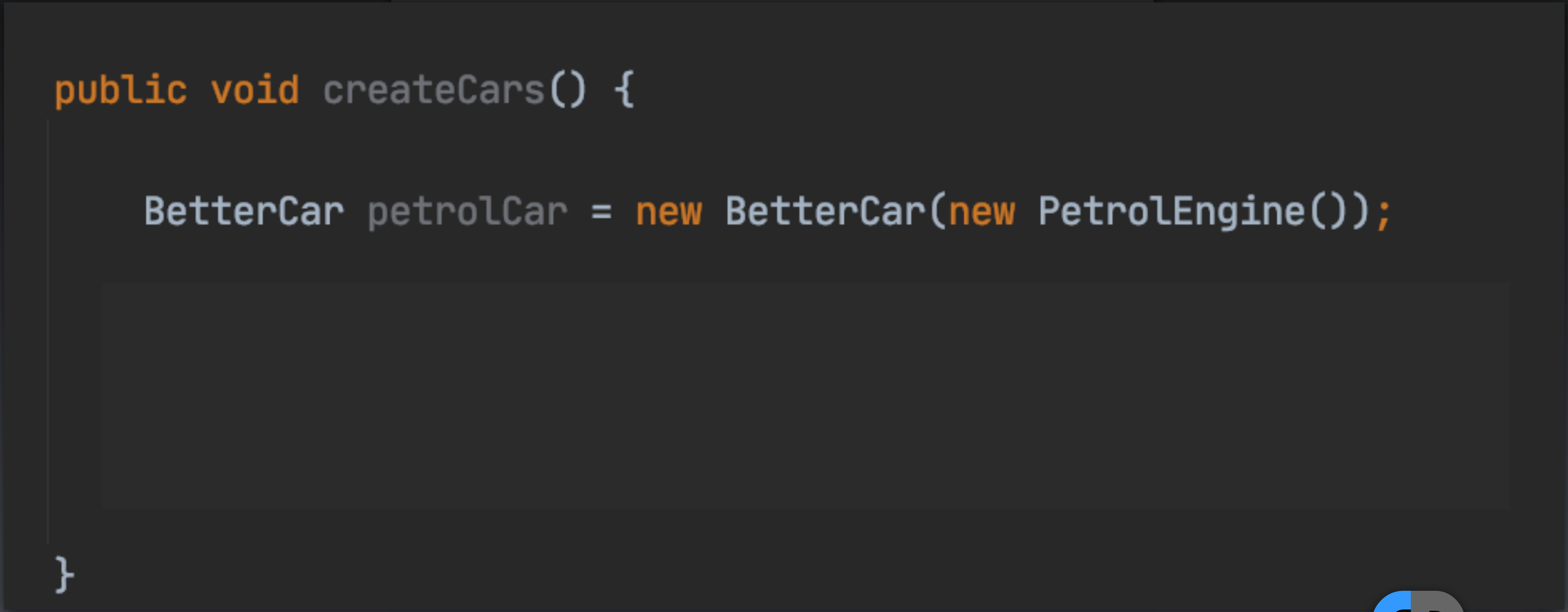
```java
public void createCars() {

}
```

Continuous Delivery ltd

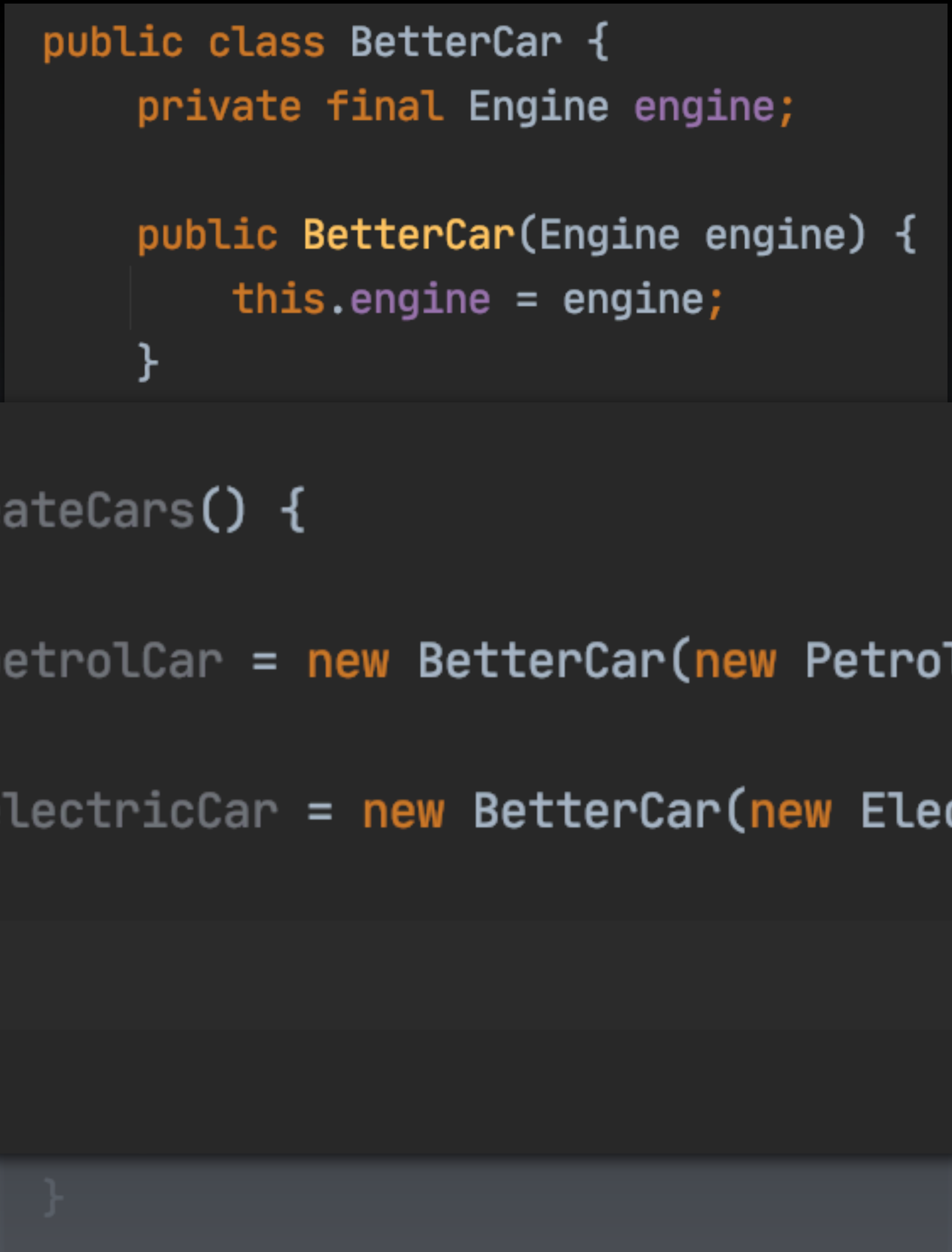


```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```

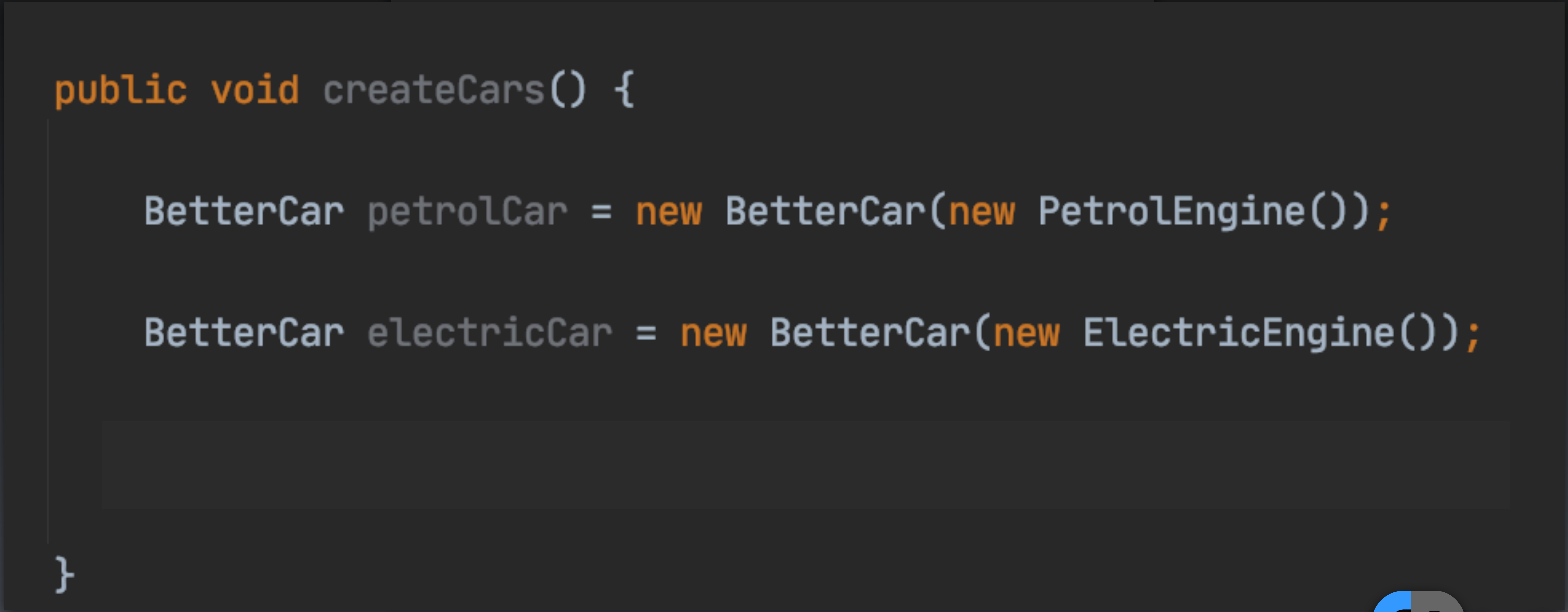
```java
}
```



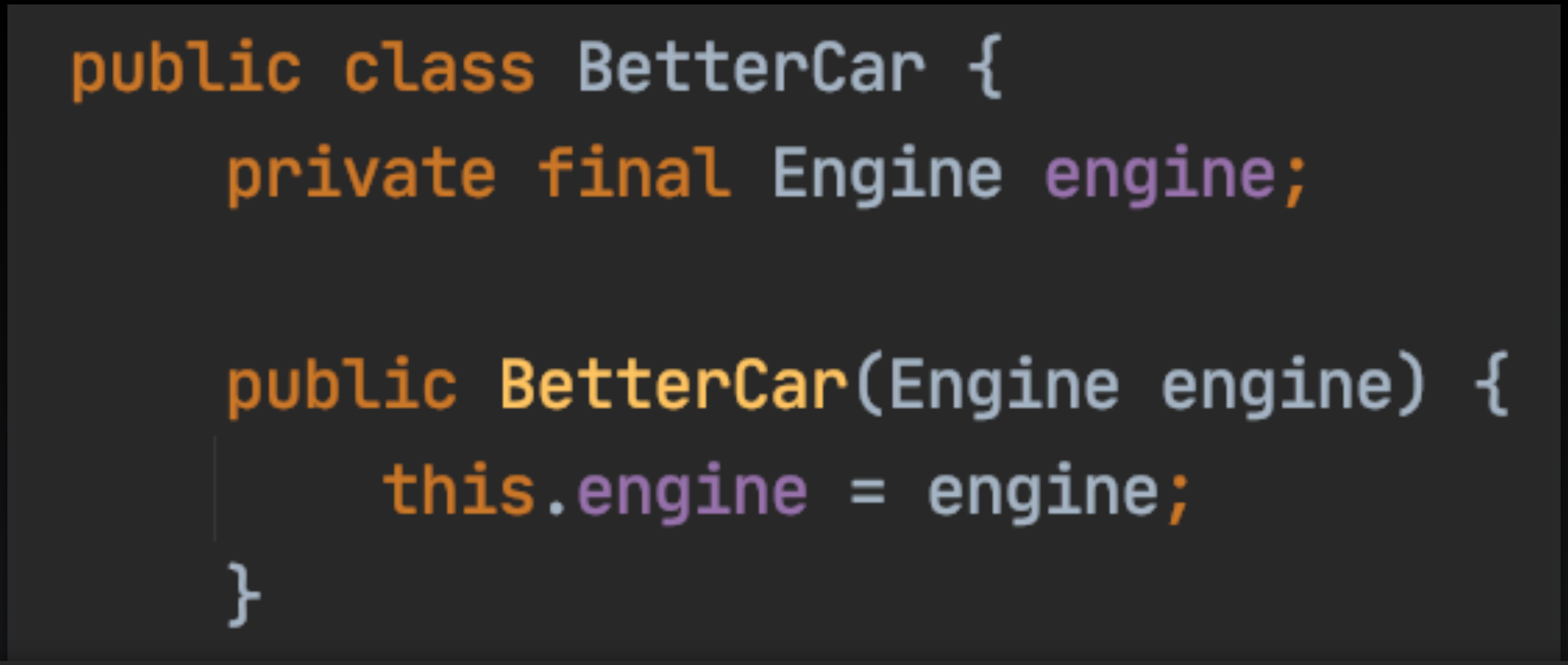
```java
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

}
```



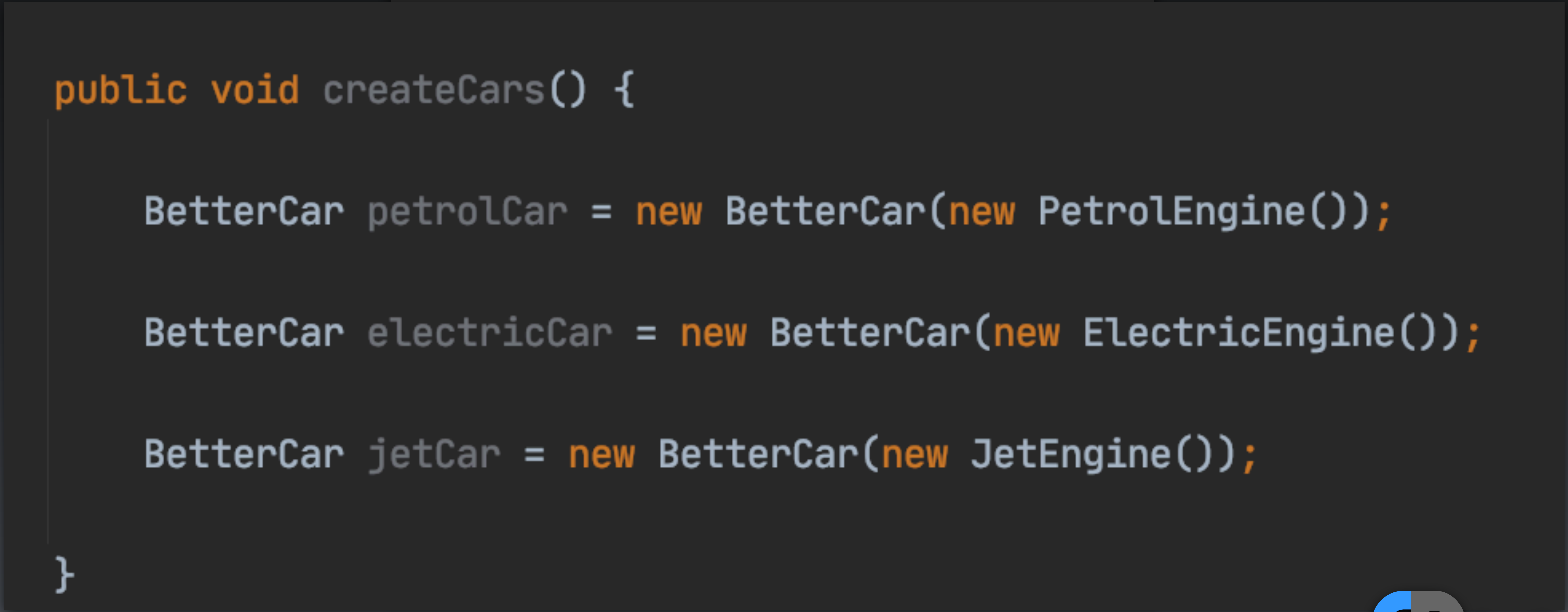
```java
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
}
```



```java
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

    BetterCar electricCar = new BetterCar(new ElectricEngine());


}
```



```
public class BetterCar {
    private final Engine engine;

    public BetterCar(Engine engine) {
        this.engine = engine;
    }
```



```
public void createCars() {

    BetterCar petrolCar = new BetterCar(new PetrolEngine());

    BetterCar electricCar = new BetterCar(new ElectricEngine());

    BetterCar jetCar = new BetterCar(new JetEngine());

}
```

The Tools of Our Trade!

Optimise for Learning	Optimise to Manage Complexity
Iteration	Modularity
Feedback	Cohesion
Incremental	Separation of Concerns
Experimental	Abstraction
Empirical	Coupling

Continuous Delivery ltd

- ***These Things Matter for 1 Reason***

- *These Things Matter for 1 Reason*
- *They Allow us to Change Our Software*

Embrace Change!

Continuous
Delivery ltd

Embrace Change!

JS Frameworks Change All the Time!

Requirements Change

Technologies Change

Businesses Change

People Change

Plans Change

Markets Change

Our Designs Change

Customers Change

Our Understanding Changes

The Quality of a System
is Defined by
Our Ability to Change it!

Continuous
Delivery ltd

Q&A

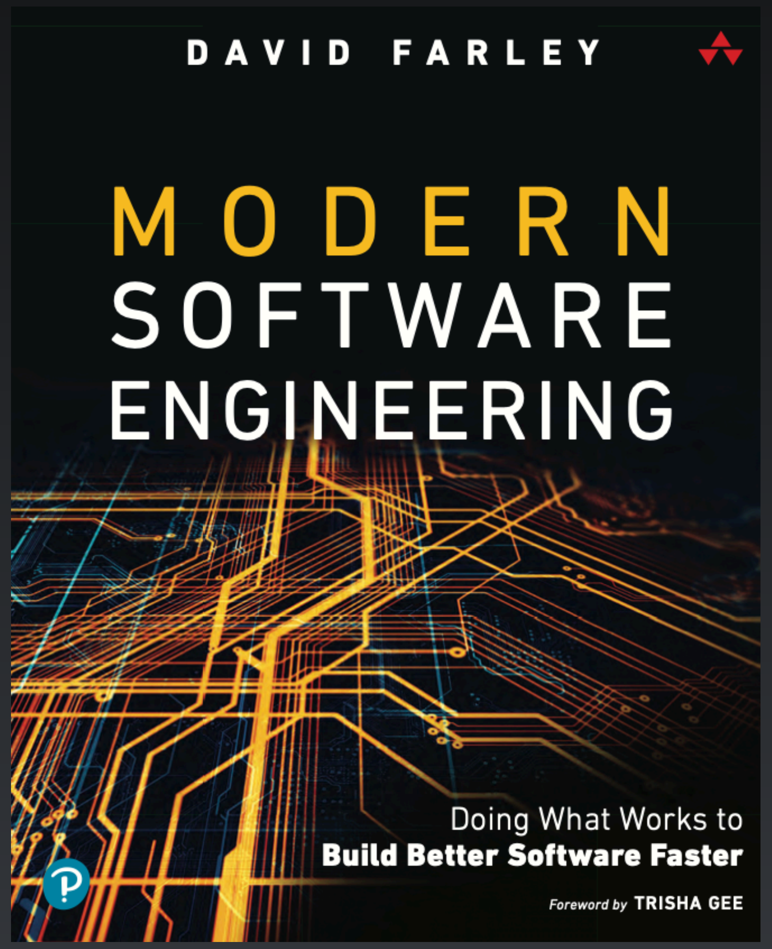
http://www.continuous-delivery.co.uk

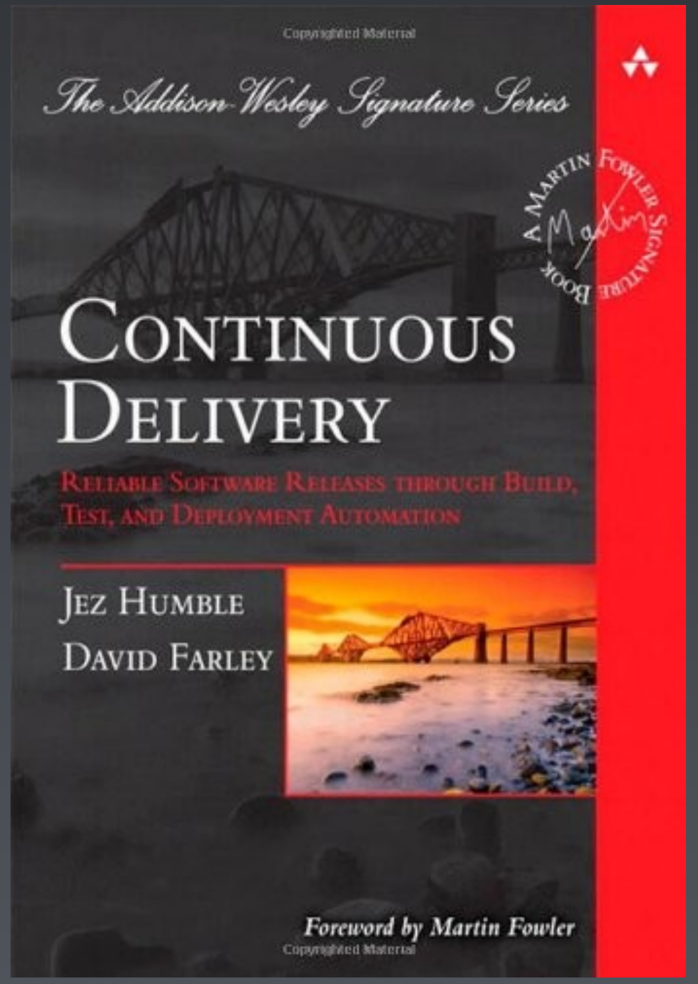
Dave Farley

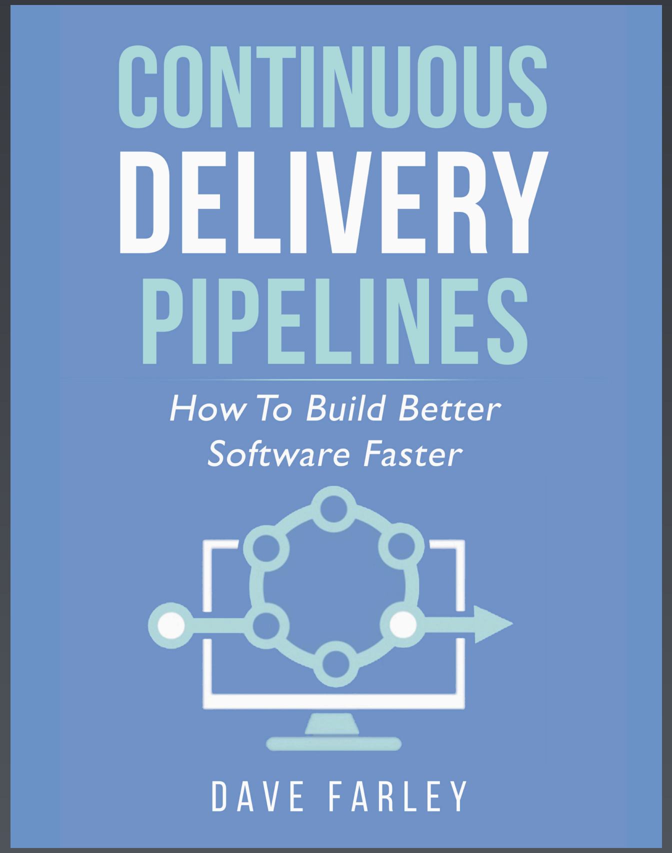
https://www.davefarley.net

@davefarley77

https://bit.ly/CDonYT


DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to Build Better Software Faster
Foreword by TRISHA GEE


The Addison-Wesley Signature Series
CONTINUOUS DELIVERY
Reliable Software Releases through Build, Test, and Deployment Automation
Jez Humble
David Farley
Foreword by Martin Fowler


CONTINUOUS DELIVERY PIPELINES
How To Build Better Software Faster
DAVE FARLEY