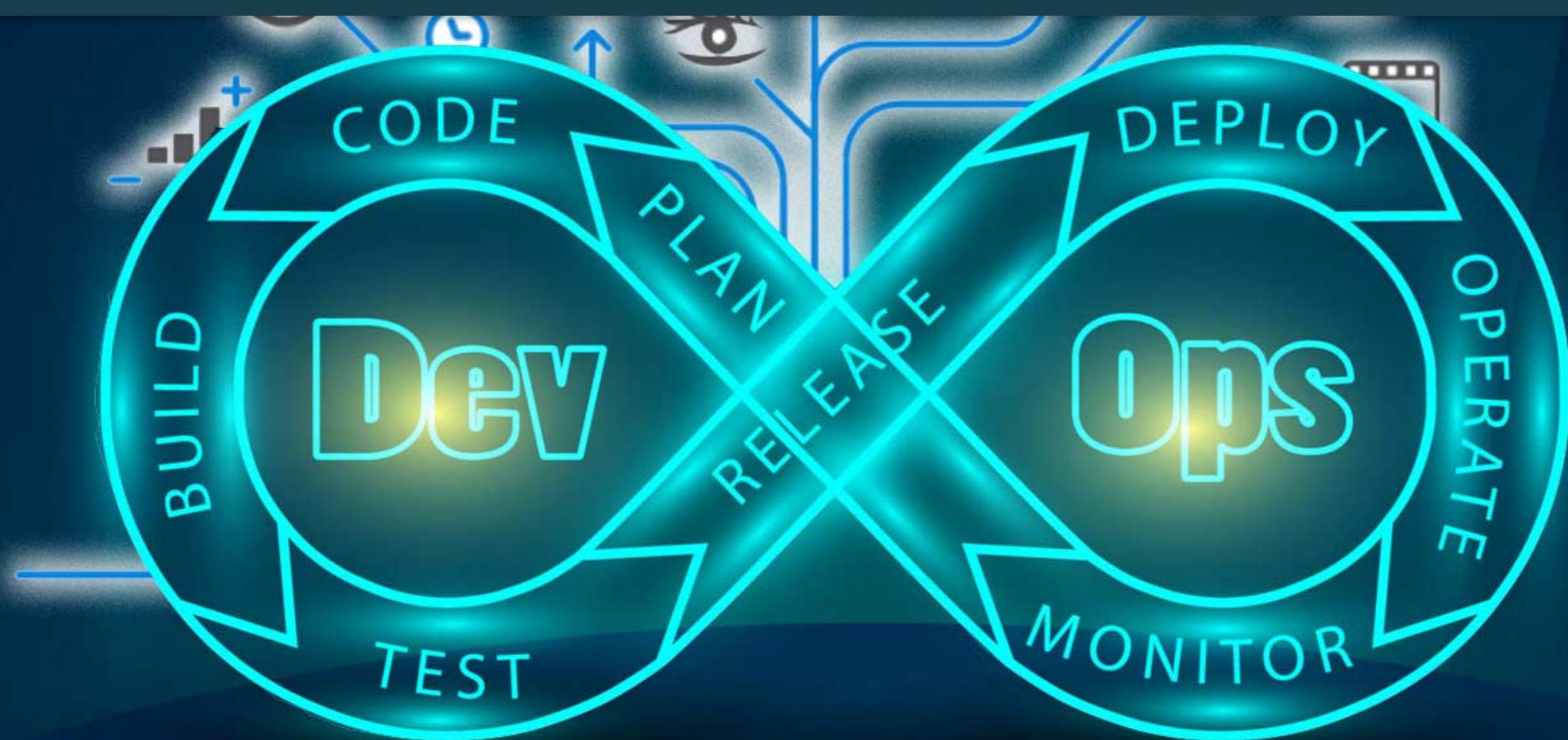
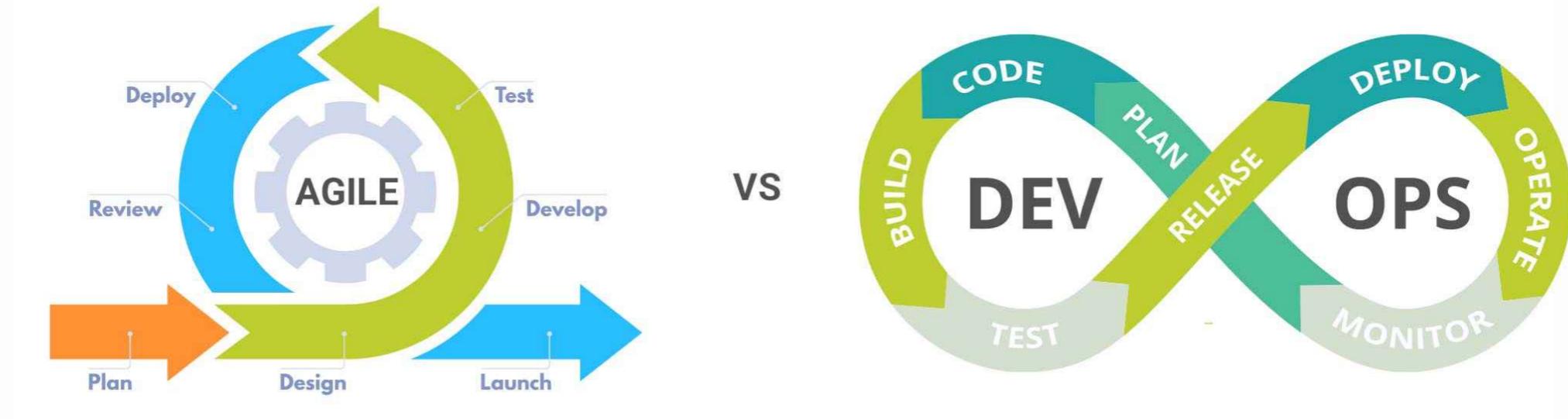


Ingegneria, Gestione ed Evoluzione del Software

DevOps

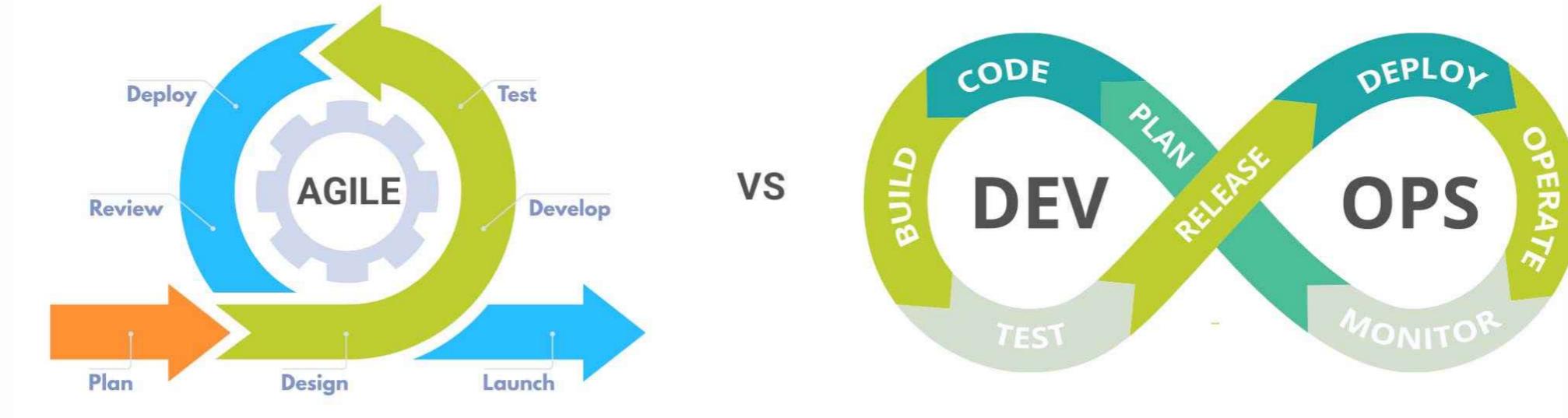


Un po' di contesto su DevOps



DevOps è il risultato **dell'applicazione dei principi più affidabili** provenienti dal campo della **produzione fisica e della leadership al flusso di valore IT**.

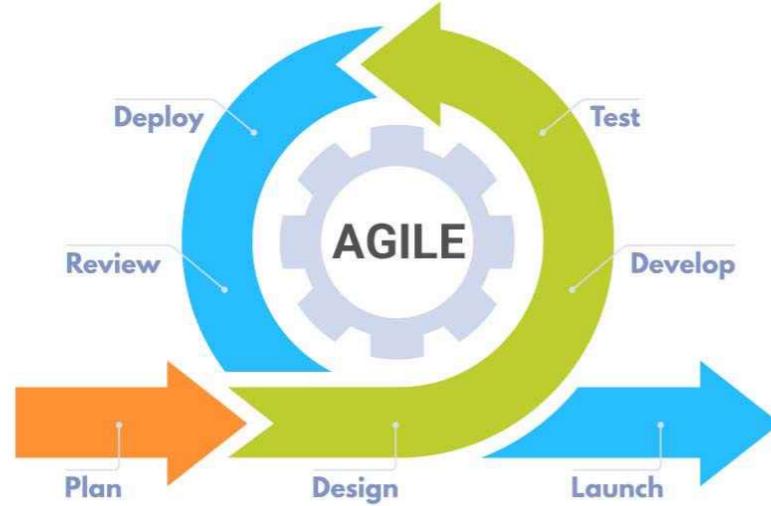
Un po' di contesto su DevOps



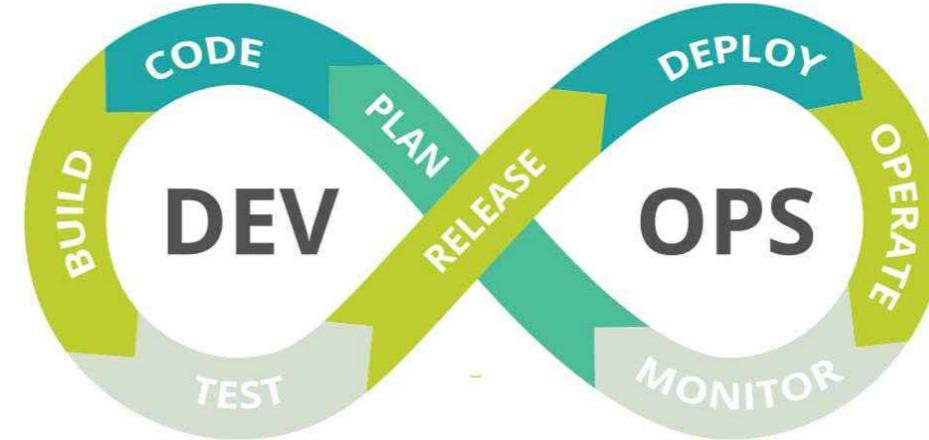
DevOps è il risultato **dell'applicazione dei principi più affidabili** provenienti dal campo della **produzione fisica e della leadership al flusso di valore IT**.

DevOps si basa su **body of knowledge** provenienti da **Lean, Theory of Constraints, il Sistema di Produzione Toyota**, ingegneria della resilienza, organizzazioni di apprendimento, cultura della sicurezza, fattori umani e molti altri.

Un po' di contesto su DevOps



VS

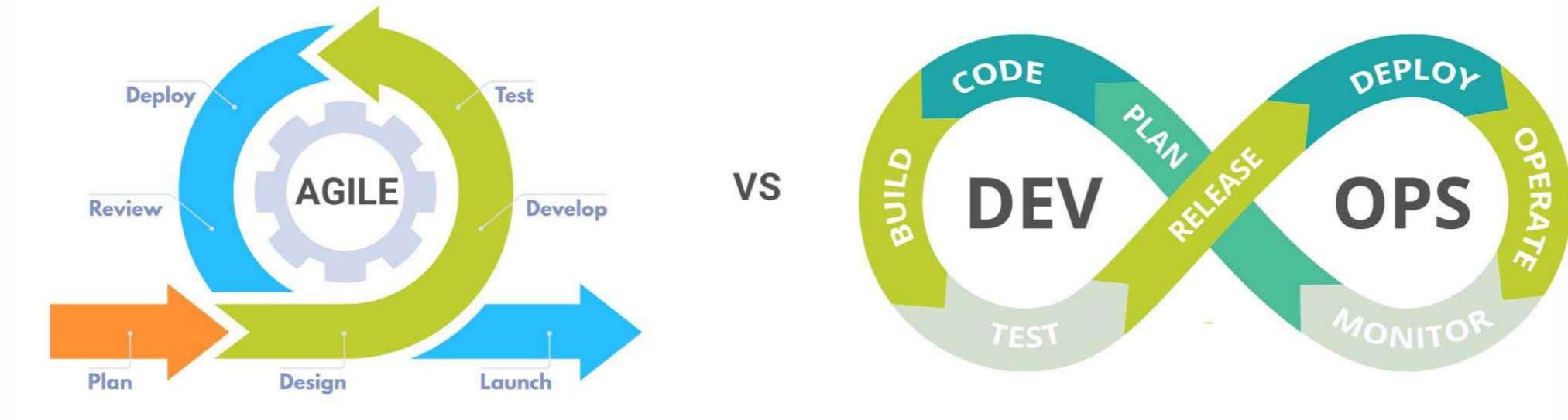


DevOps è il risultato **dell'applicazione dei principi più affidabili** provenienti dal campo della **produzione fisica e della leadership al flusso di valore IT**.

DevOps si basa su **body of knowledge** provenienti da **Lean, Theory of Constraints, il Sistema di Produzione Toyota**, ingegneria della resilienza, organizzazioni di apprendimento, cultura della sicurezza, fattori umani e molti altri.

Il principio chiave è "**consegnare frequentemente software funzionante, da un paio di settimane a un paio di mesi, con una preferenza per il periodo più breve**", sottolineando il desiderio di dimensioni di lotto ridotte, rilasci incrementali invece di rilasci grandi secondo l'approccio waterfall.

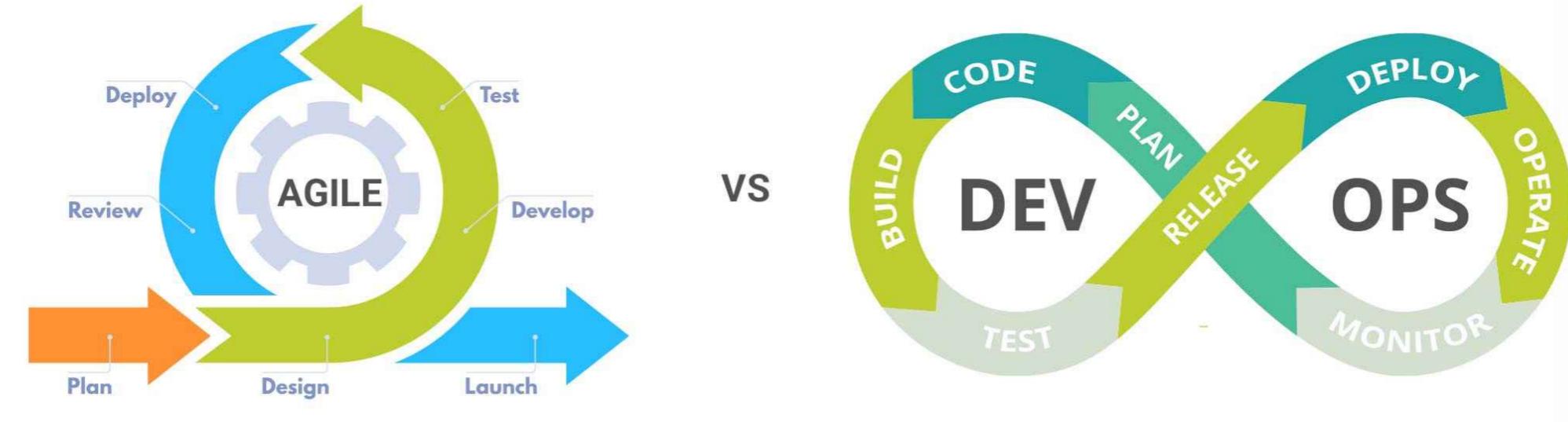
Un po' di contesto su DevOps



Value Stream (Flusso di valore): “La sequenza di attività necessarie per **progettare, produrre e consegnare un bene o un servizio** a un cliente, inclusi i flussi duali di informazioni e materiali.”

In DevOps, discutiamo il nostro flusso di valore tecnologico come il flusso necessario per trasformare un'ipotesi aziendale in un servizio abilitato dalla tecnologia che offre valore al cliente.

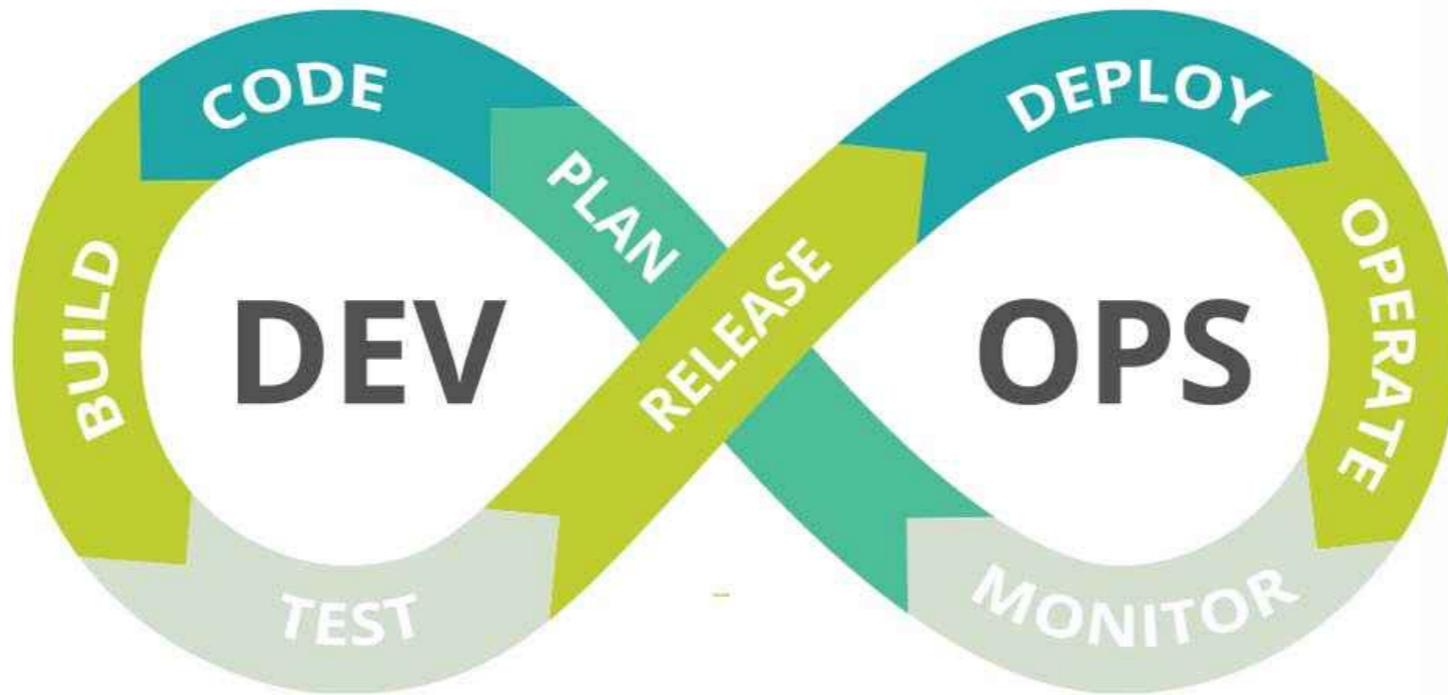
Un po' di contesto su DevOps



Value Stream (Flusso di valore): “La sequenza di attività necessarie per **progettare, produrre e consegnare un bene o un servizio** a un cliente, inclusi i flussi duali di informazioni e materiali.”

In DevOps, di solito definiamo il nostro flusso di valore tecnologico come il **processo necessario per trasformare un'ipotesi aziendale in un servizio abilitato dalla tecnologia** che offre valore al cliente.

Perché DevOps

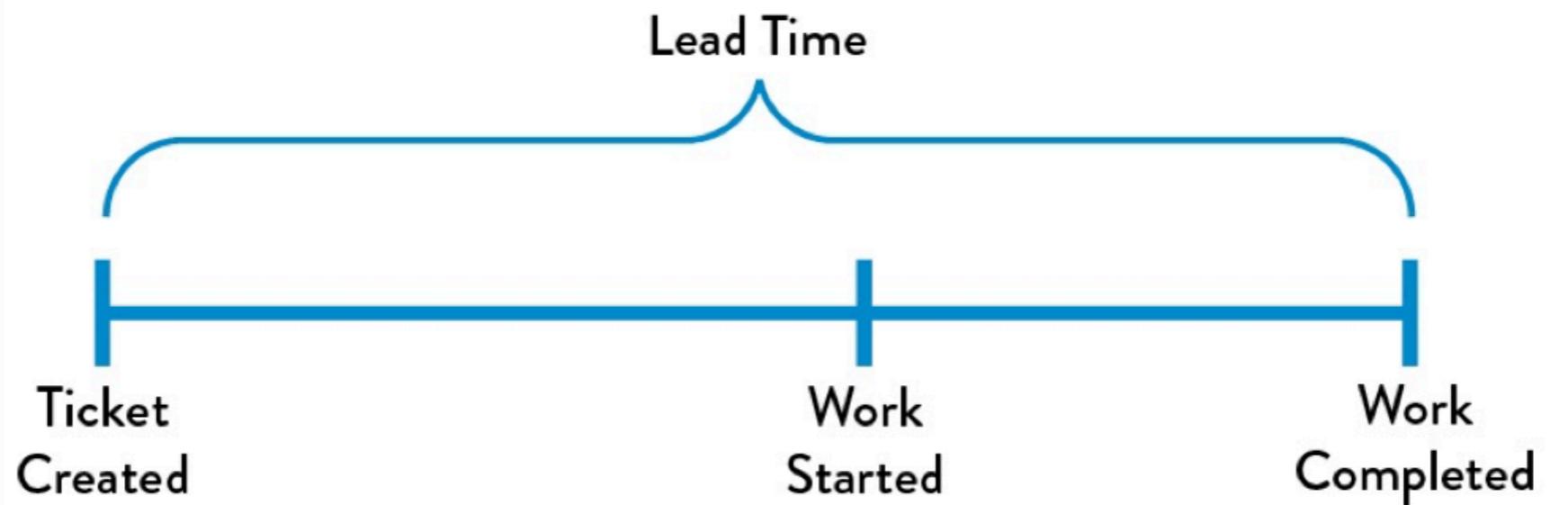


DevOps punta a **minimizzare il value stream**.

A questo punto definiamo:

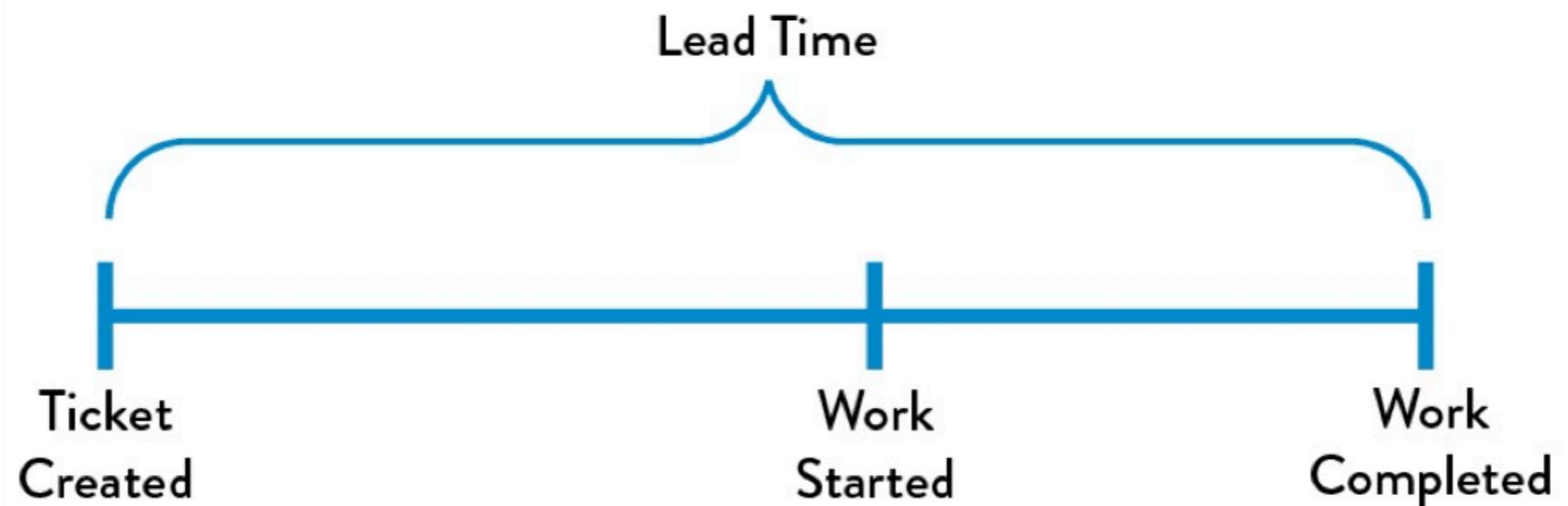
“DevOps è un insieme di pratiche volte a ridurre il tempo che intercorre tra il commit di una modifica a un sistema e la modifica nella produzione normale, garantendo al contempo un'elevata qualità.”

Perché DevOps



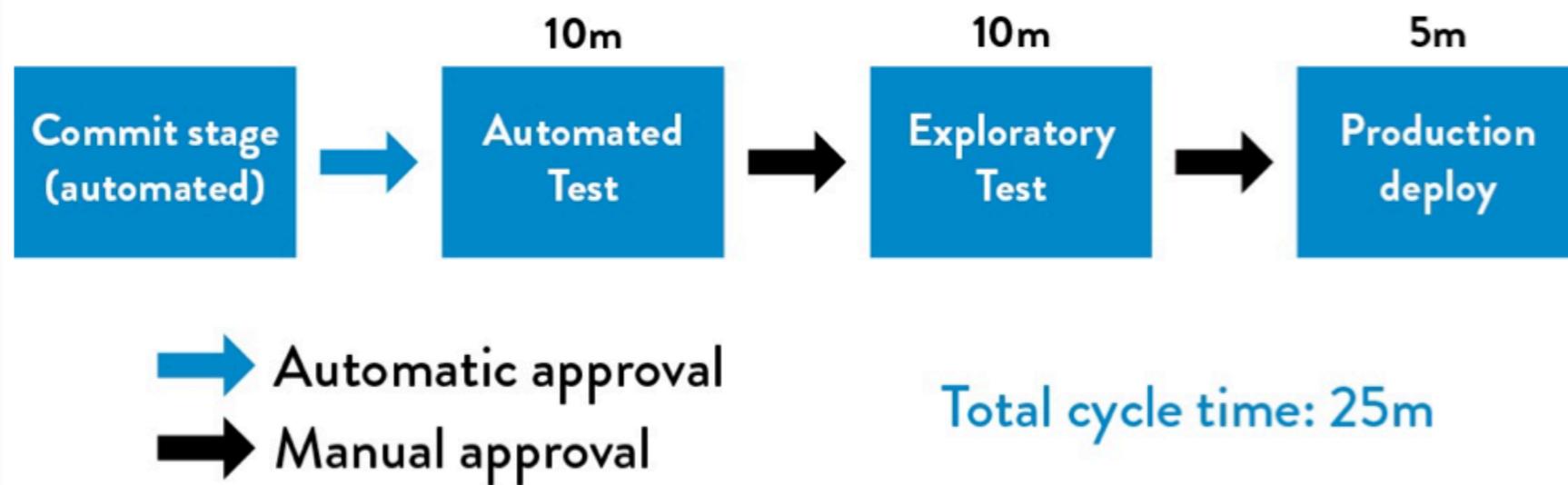
Lead Time: Tempo che intercorre tra la creazione di una richiesta e il completamento di un lavoro.

Perché DevOps



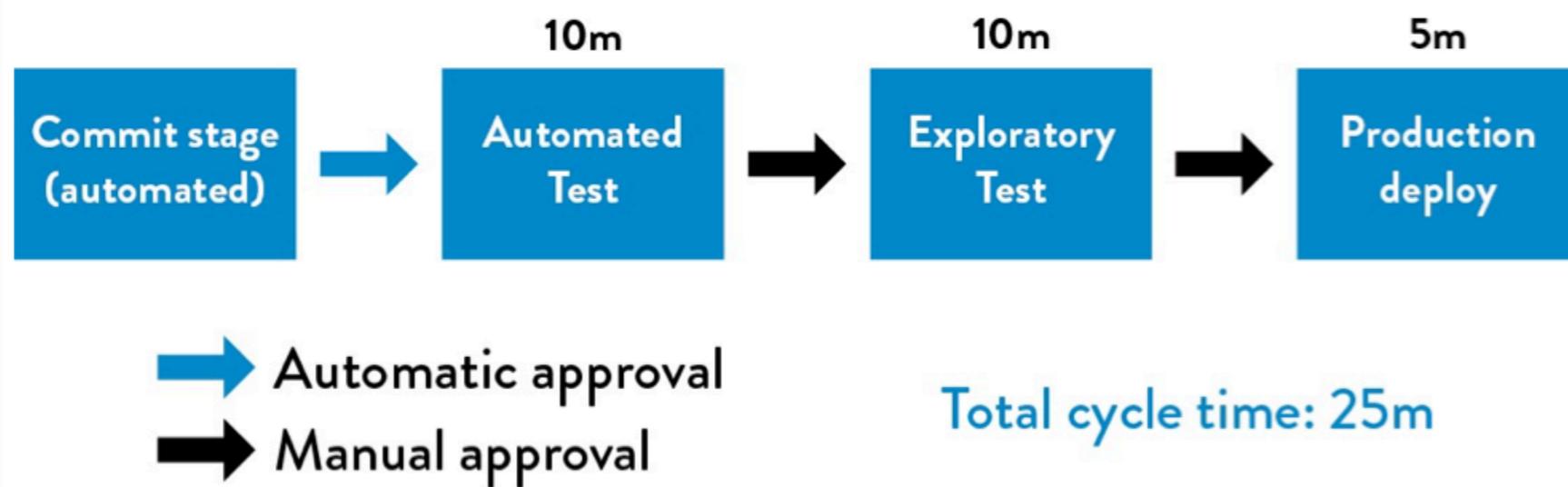
Lead Time: Tempo che intercorre tra la creazione di una richiesta e il completamento di un lavoro.

Scenario Ideale di DevOps: minimizzare parte del lead time in termini di minuti!



Problema!

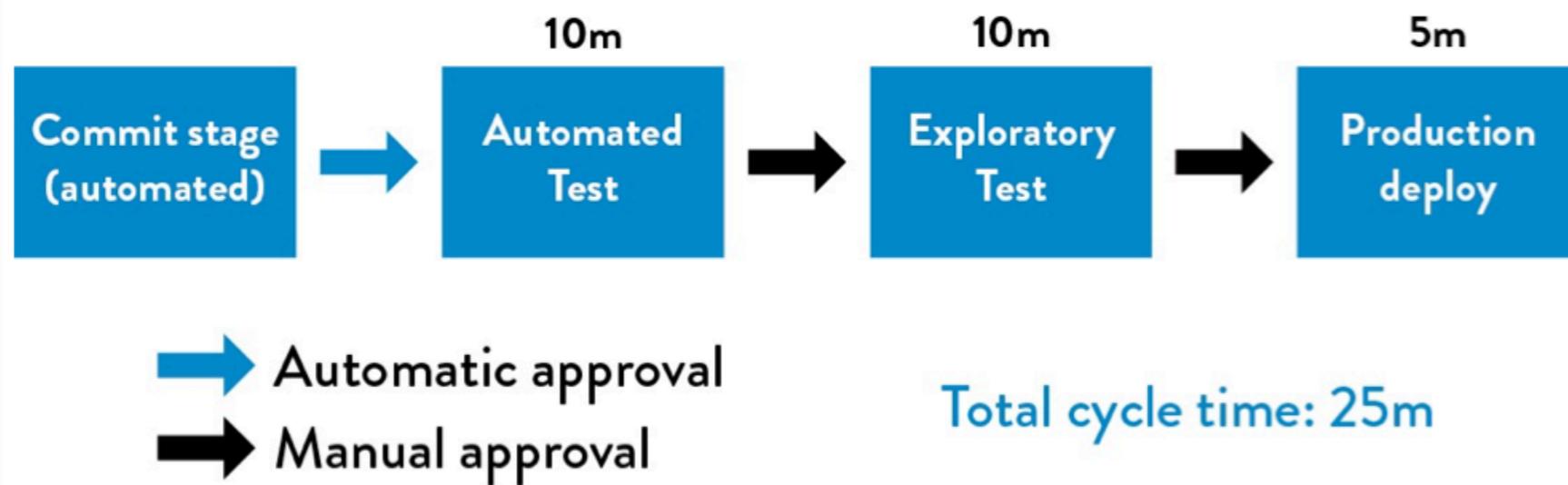
L
U
S



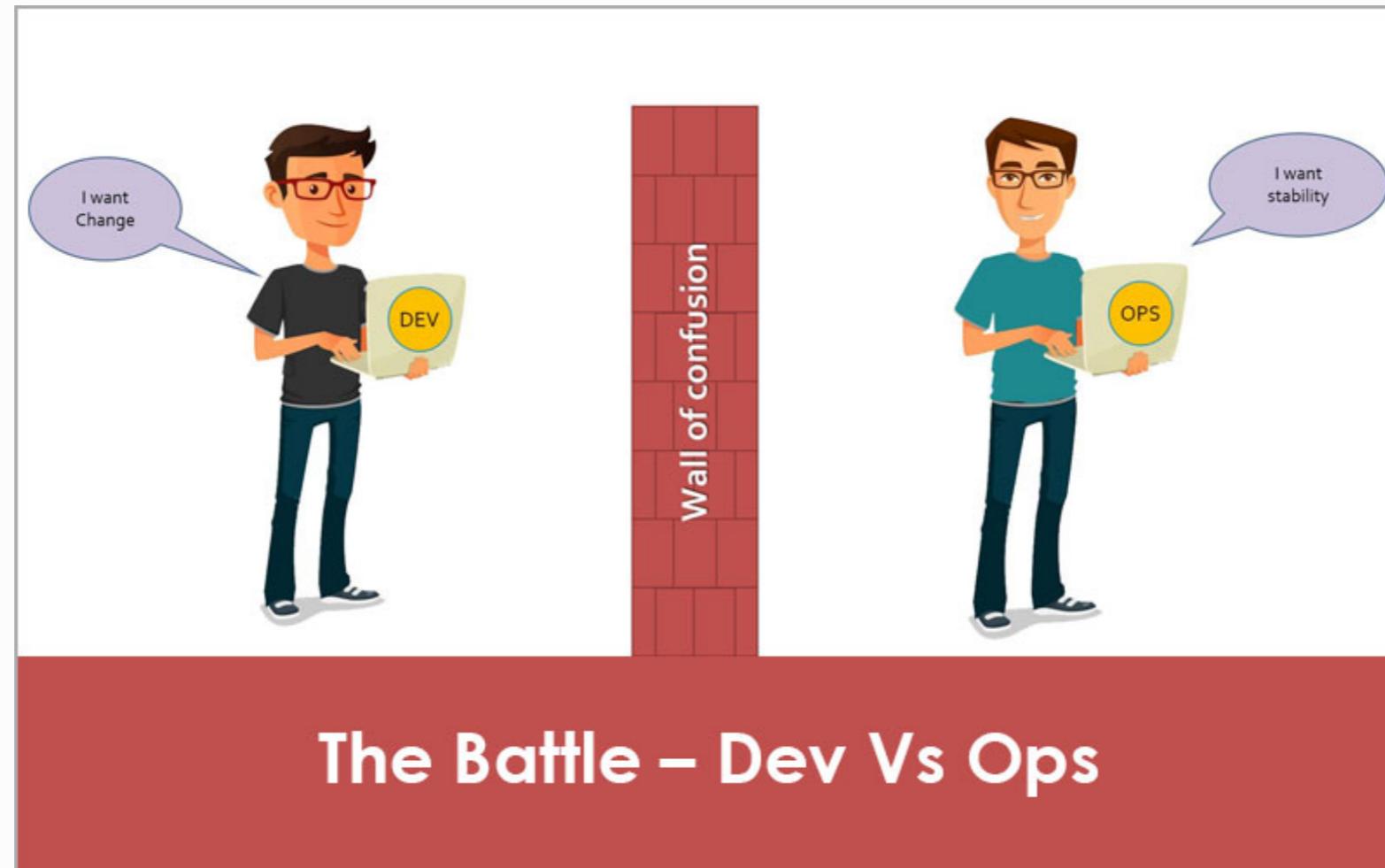
Problema!

Le fasi che vediamo qui sotto non coinvolgono solo il ciclo di vita relativo allo sviluppo del software.

Per poter raggiungere questo obiettivo, DevOps deve interagire sia sulla fase di Development che su quella di Operations!



Perché DevOps



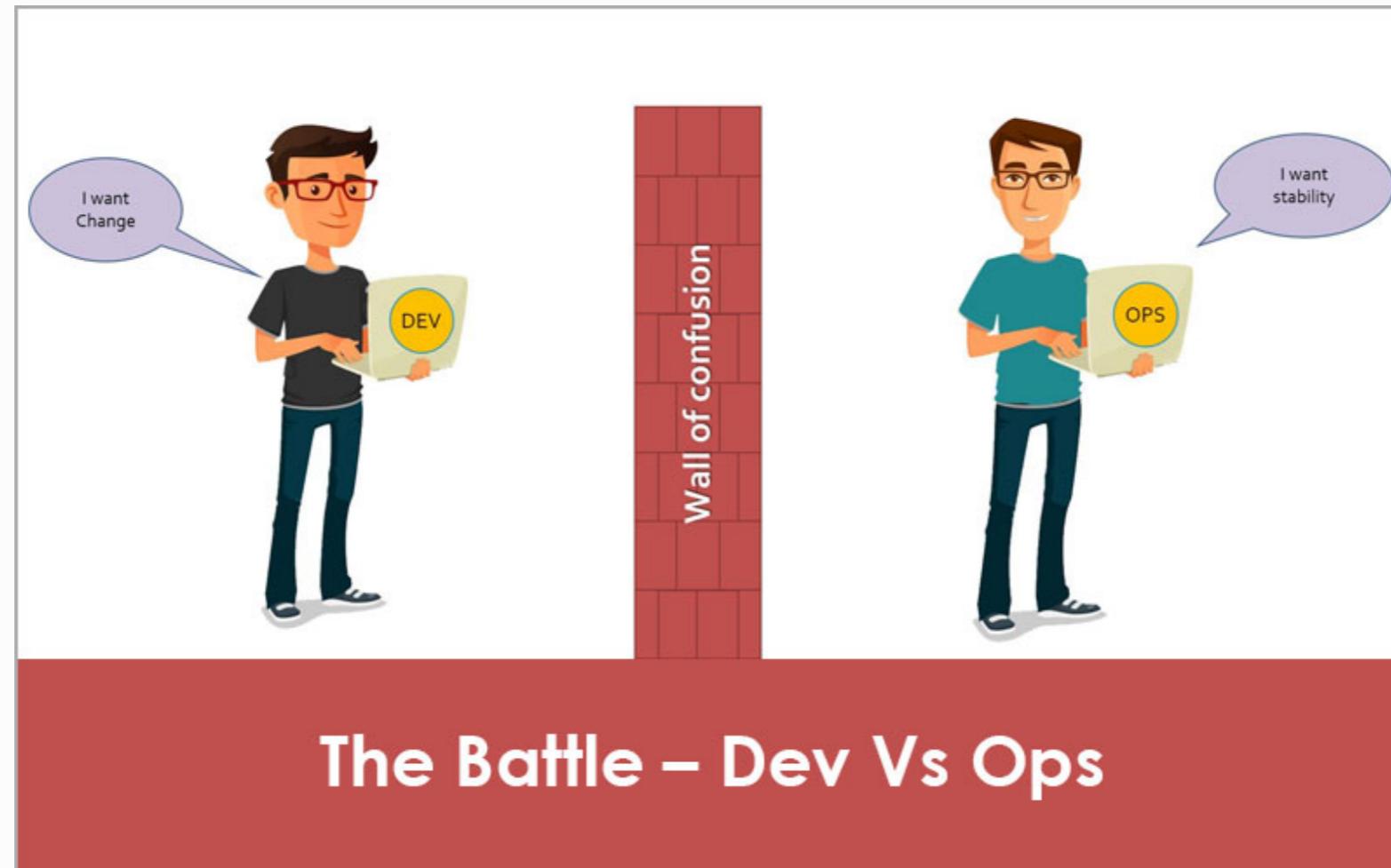
Dev

- Il team Dev punta a fornire **nuove versioni del software**.
- Il team Dev **non** presta attenzione al Quality of Service.

Ops

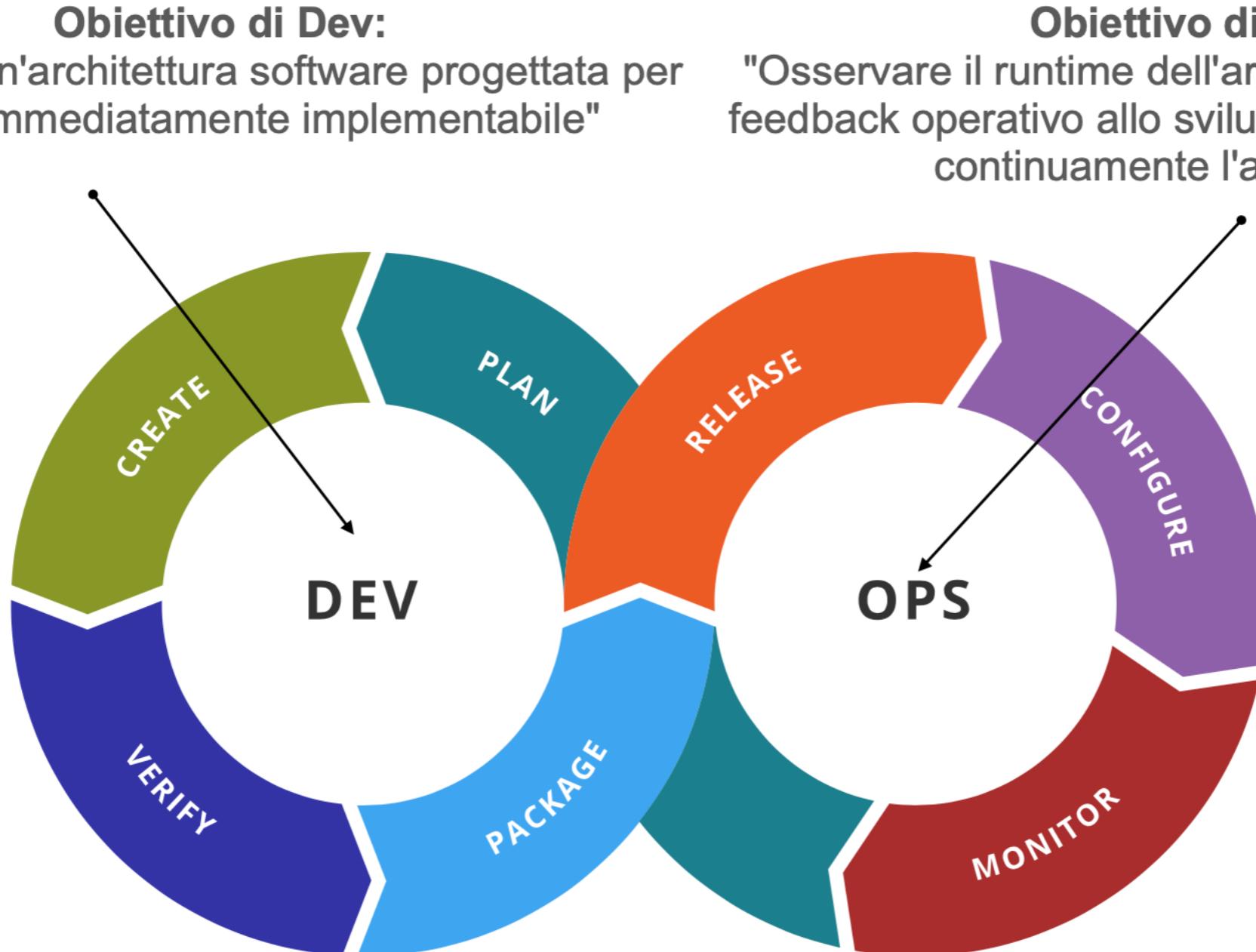
- Il team Ops punta a **ritardare la messa in produzione** di nuove versioni del software.
- Il team Ops **punta** a garantire il Quality of Service.

Perché DevOps



DevOps fornisce pratiche e strumenti che colmano il divario tra Dev ed Ops e creare una realtà collaborativa tra i due team.

Perché DevOps



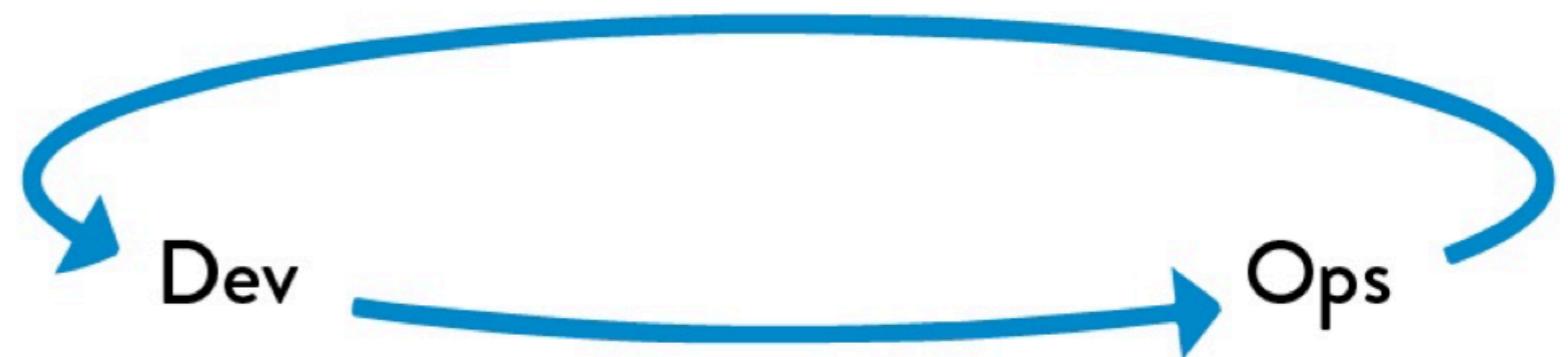
The Three Ways of DevOps

The Phoenix Project[1] definisce I principi su cui si basano tutte le pratiche e gli strumenti che ci permettono di applicare DevOps.

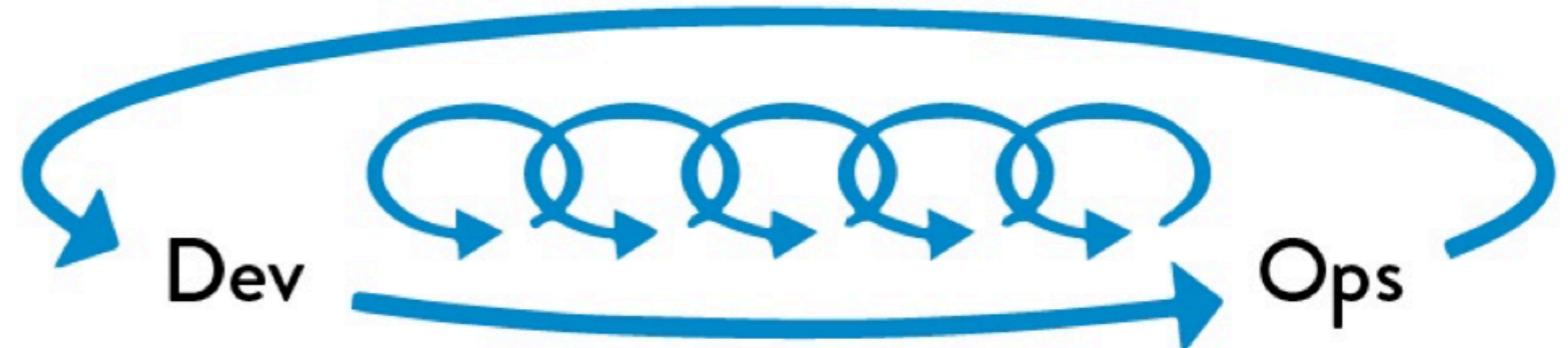
1: Fast left-to-right flow of work



2: Fast and constant flow of feedback



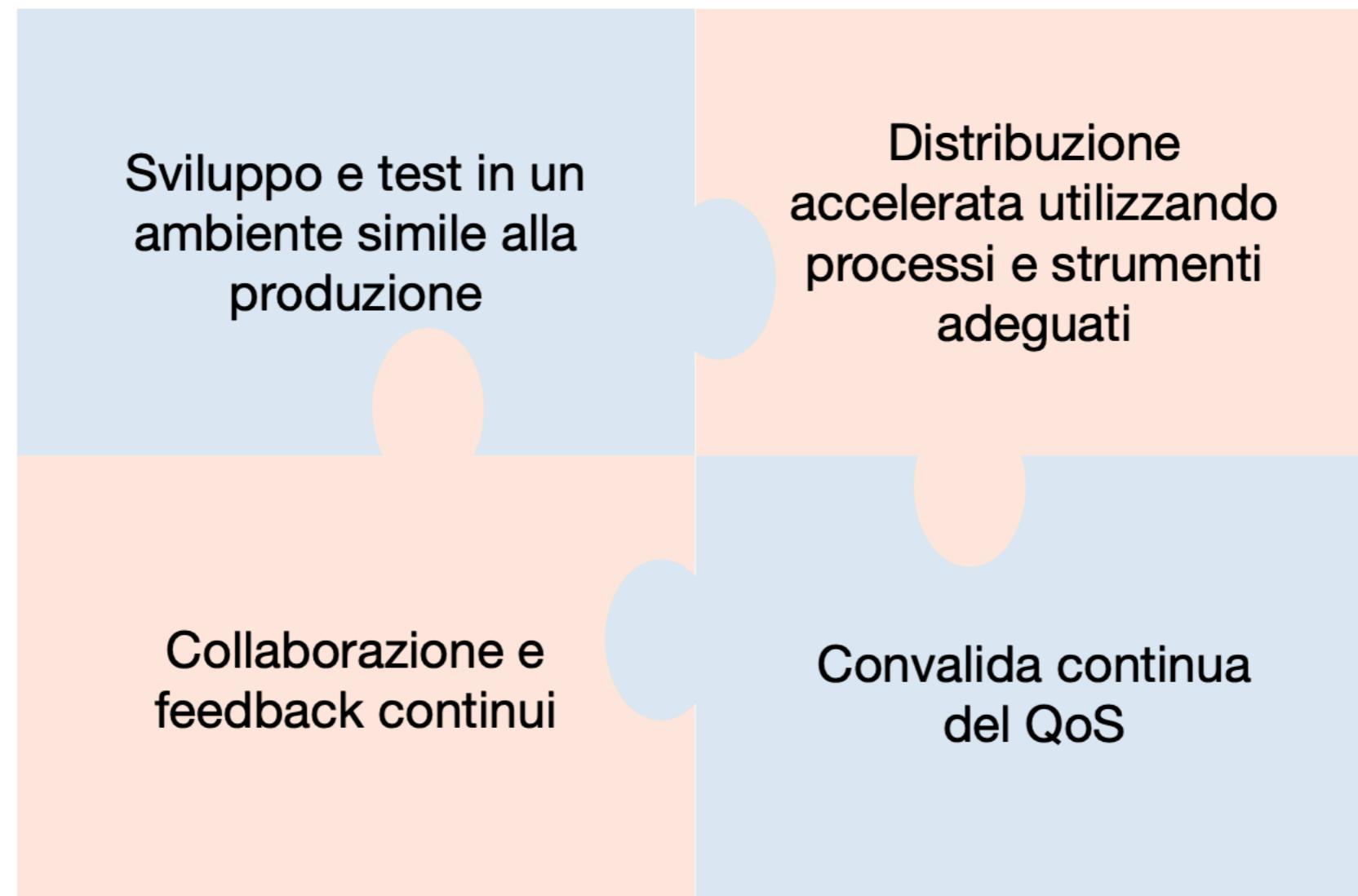
3: Continual Learning and Experimentation



[1] Gene Kim, Kevin Behr, and George Spafford. 2013. *The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win* (1st. ed.). IT Revolution Press.

The Three Ways of DevOps

Per garantire i principi di DevOps abbiamo bisogno quindi di:



The Three Ways of DevOps

Per garantire i principi di DevOps abbiamo bisogno quindi di:

Continuous Architecting

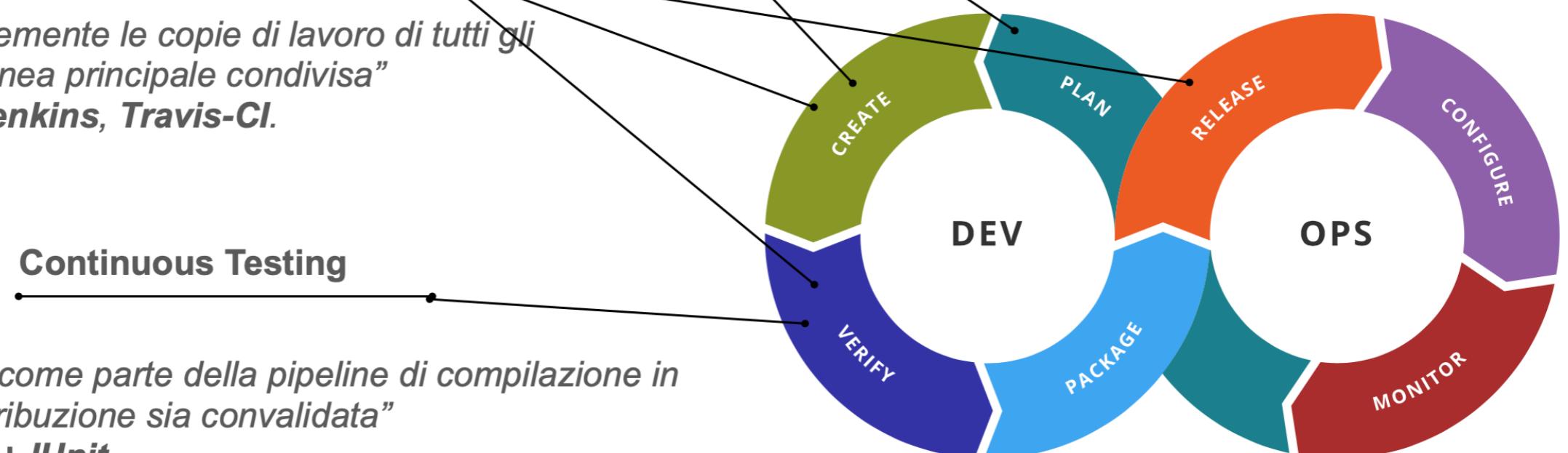
Def. “architettare per testare, costruire e distribuire, prendendo in considerazione gli attributi di qualità e sfruttando il feedback dal runtime”

Continuous Integration

Def. “unire frequentemente le copie di lavoro di tutti gli sviluppatori in una linea principale condivisa”
Esempi. Apache Jenkins, Travis-CI.

Continuous Testing

Def. “eseguire test come parte della pipeline di compilazione in modo che ogni distribuzione sia convalidata”
Esempi. Selenium+JUnit.



Continuous Architecting

L'obiettivo della continuous architecting è quello di garantire che l'architettura del sistema si evolva e si adatti continuamente per soddisfare le esigenze aziendali e tecnologiche in un ambiente di sviluppo e operativo in continua evoluzione.

Ciò include la creazione di un'architettura **modulare, flessibile e scalabile** che consenta la **distribuzione continua del software** e delle **modifiche** nel contesto DevOps.

Le funzionalità chiave della continuous architecting includono:

1. **Microservices:** Creazione di architetture basate su **Microservizi**.
2. **Containerization:** Creazione di **container isolati** per provare i cambiamenti **della parte di Dev** in ambienti Ops simulati che siano il **più simili possibile alla produzione**.
3. **IaC:** Definizione dell'**infrastruttura del sistema come codice** (**Infrastructure as Code**), consentendo una gestione più efficiente, riproducibile e scalabile delle risorse di infrastruttura.
4. **Configuration Management Tools:** Configurazione gestione automatica delle componenti del sistema (o Orchestrazione), garantendo che l'architettura sia coerente e correttamente configurata in tutti gli ambienti.

Continuous Integration

La continuous integration (CI) è una pratica di sviluppo del software che coinvolge **l'integrazione regolare e automatica del codice di sviluppo in un repository condiviso**.

L'obiettivo principale della CI è quello di **ridurre i problemi di integrazione, identificare errori precocemente** come parte di un processo di sviluppo collaborativo e continuo.

Le funzionalità chiave della continuous integration includono:

1. **Automazione delle build:** La CI consente di **automatizzare il processo di creazione dell'eseguibile**, dei pacchetti o del deploy del software. Ciò assicura che il codice sorgente venga compilato in modo coerente e senza errori.
2. **Analisi statica del codice:** La CI può eseguire automaticamente analisi statica del codice per identificare **potenziali problemi, come violazioni di stile, errori comuni o vulnerabilità di sicurezza**.
3. **Notifiche e feedback tempestivi:** La CI **fornisce notifiche immediate** agli sviluppatori sui risultati delle build e dei test. In caso di errori o fallimenti, vengono inviate notifiche per consentire agli sviluppatori di risolvere rapidamente i problemi.
4. **Reportistica e monitoraggio:** La CI **genera report sullo stato delle build**, dei test e delle metriche di qualità del codice, che possono essere consultati per valutare la salute del progetto e l'avanzamento del lavoro.
5. **Versionamento del codice:** La CI tiene traccia delle **diverse versioni del codice sorgente**, consentendo di **ripristinare versioni precedenti** in caso di necessità.

Continuous Testing

Il continuous testing (test continuo) è una pratica che si integra nella Continuous Integration (CI) nella pipeline di sviluppo e delivery del software all'interno di un approccio DevOps.

Consiste nell'**esecuzione automatica e continua dei test** durante tutto il ciclo di vita del software, in modo da identificare tempestivamente eventuali problemi o regressioni e garantire che il software funzioni come previsto.

Le funzionalità principali del continuous testing includono:

1. **Automazione dei test:** Il continuous testing prevede l'automazione dei test, compresi i test unitari, i test di integrazione, i test funzionali, i test di regressione e altri tipi di test. Ciò permette di eseguire i test in modo rapido, ripetibile e affidabile.
2. **Integrazione con la pipeline di CI/CD:** Il continuous testing viene integrato nella pipeline di continuous integration e continuous delivery. Ciò significa che i test vengono eseguiti automaticamente ogni volta che viene effettuata una nuova build o viene apportata una modifica al codice.
3. **Esecuzione dei test in ambienti simili alla produzione:** I test vengono eseguiti in ambienti che sono il più possibile simili a quello di produzione, garantendo una maggiore affidabilità dei risultati dei test e riducendo i rischi di problemi che potrebbero verificarsi solo in ambienti diversi.
4. **Test paralleli e distribuiti:** Il continuous testing può essere **eseguito in parallelo** su più ambienti o su diverse istanze di test per accelerare i tempi di esecuzione e ottenere risultati più rapidi.

The Three Ways of DevOps

Per garantire i principi di DevOps abbiamo bisogno quindi di:

1 Fm Gh Github	3 Os Gt Git	4 En Dm DBmaestro	2 Fm Bb Bitbucket	12 Os Lb Liquibase	11 Fm Gl GitLab	20 En Rg Redgate	21 Os Mv Maven	22 Os Gr Gradle	23 Os At ANT	24 Os Fn FitNesse	25 Fr Se Selenium	26 Os Ga Gatling	27 Fr Dh Docker Hub	28 Os Jn Jenkins	29 Pd Ba Bamboo	30 Os Tr Travis CI	31 Pd Gd Deployment Manager	32 Os Sf SmartFrog	33 Os Cn Consul	34 Os Bc Bcfg2	35 Os Mo Mesos	36 En Rs Rackspace											
39 Os Dt Datical	40 Os Gt Grunt	41 Os Gp Gulp	42 Fr Br Broccoli	43 Os Cu Cucumber	44 Fr Cj Cucumber.js	45 Os Qu Qunit	46 Fm Npm npm	47 Pd Cs Codeship	48 Fm Vs Visual Studio	49 Fr Cr CircleCI	50 Fr Cp Capistrano	51 Os Ju JuJu	52 Os Rd Rundeck	53 Fr Cf CFEngine	54 Os Ds Swarm	55 Os Hg Mercurial	56 En Dp Delphix	57 Fr Sb sbt	58 Os Mk Make	59 Os Ck CMake	60 Fr Jt JUnit	61 Fr Jm JMeter	62 Fr Tn TestNG	63 Os Ay Artifactory	64 Fm Tc TeamCity	65 Fm Sh Shippable	66 Os Cc CruiseControl	67 En Ry RapidDeploy	68 Fm Cy CodeDeploy	69 En Oc Octopus Deploy	70 En No CA Nolio	71 Os Kb Kubernetes	72 Fm Hr Heroku
73 En Cw ISPW	74 En Id Idera	75 Os Msb MSBuild	76 Os Rk Rake	77 Fr Pk Packer	78 Os Mc Mocha	79 Fr Km Karma	80 Os Jm Jasmine	81 Os Nx Nexus	82 Os Co Continuum	83 Fm Ca Continua CI	84 Pd So Solano CI	85 En Xld XL Deploy	86 En Eb ElasticBox	87 Fm Dp Deploybot	88 En Ud UrbanCode Deploy	89 Os Nm Nomad	90 En Os OpenShift	91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Bm BMC Release Process	94 En Hp HP Cedar	95 En Au Automic	96 En Pl Plutora Release	97 En Sr Serena Release	98 Pd Tfs Team Foundation	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow	
106 Os Ki Kibana	107 Fm Nr New Relic	108 En Dt Dynatrace	109 Os Ni Nagios	110 Os Zb Zabbix	111 En Dd Datadog	112 Os Ei Elasticsearch	113 Fm Ad AppDynamics	114 En Sp Splunk	115 Fm Le Logentries	116 Fm Sl Sumo Logic	117 Os Ls Logstash	118 Os Sn Snort	119 Os Tr Tripwire	120 En Ff Fortify																			

XebiaLabs

Follow @xebialabs

91 En Xlr XL Release	92 En Ur UrbanCode Release	93 En Bm BMC Release Process	94 En Hp HP Cedar	95 En Au Automic	96 En Pl Plutora Release	97 En Sr Serena Release	98 Pd Tfs Team Foundation	99 Fm Tr Trello	100 Pd Jr Jira	101 Fm Rf HipChat	102 Fm Sl Slack	103 Fm Fd Flowdock	104 Pd Pv Pivotal Tracker	105 En Sn ServiceNow
106 Os Ki Kibana	107 Fm Nr New Relic	108 En Dt Dynatrace	109 Os Ni Nagios	110 Os Zb Zabbix	111 En Dd Datadog	112 Os Ei Elasticsearch	113 Fm Ad AppDynamics	114 En Sp Splunk	115 Fm Le Logentries	116 Fm Sl Sumo Logic	117 Os Ls Logstash	118 Os Sn Snort	119 Os Tr Tripwire	120 En Ff Fortify

Ingegneria, Gestione ed Evoluzione del Software

DevOps

