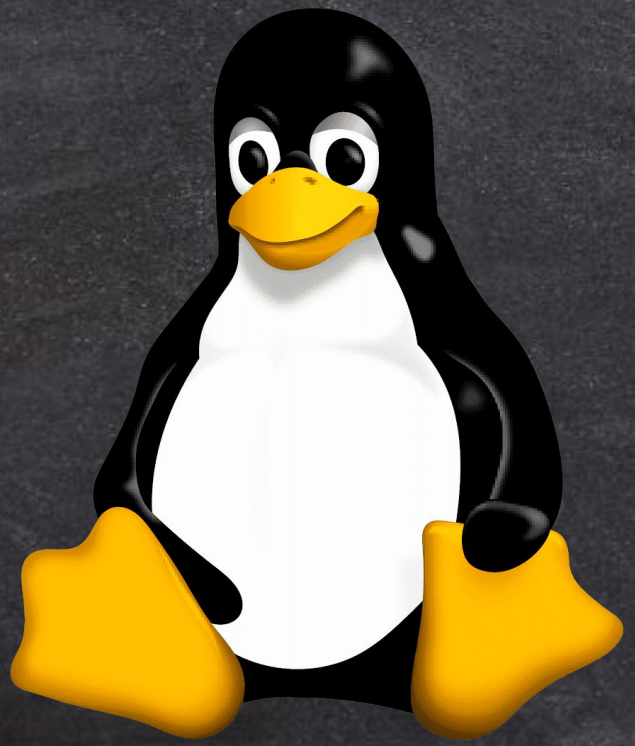
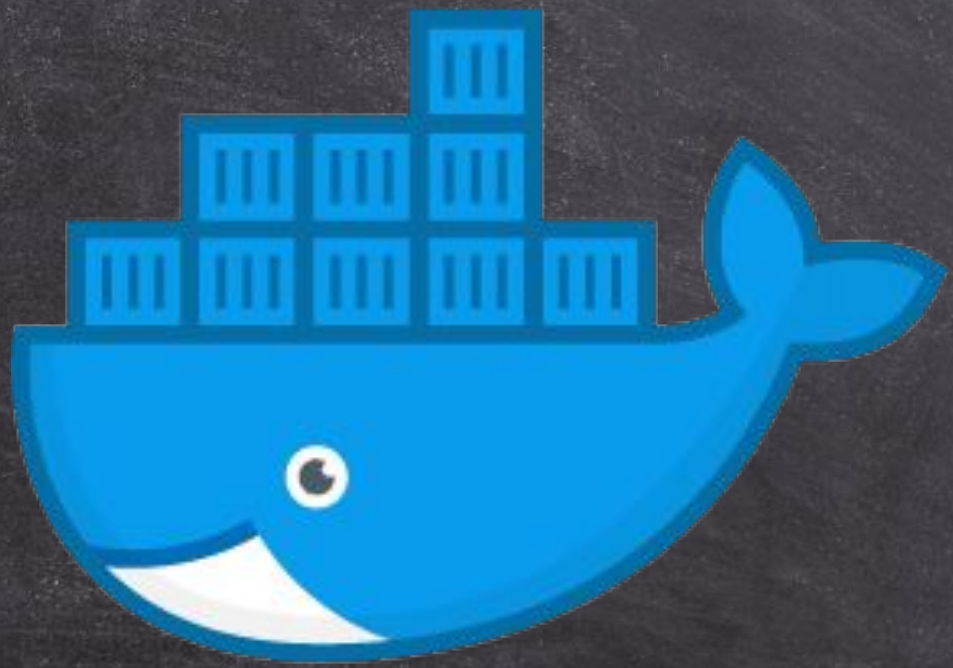
DevOps
Session #3

CI & CD
Using Jenkins

Dor Alteresku — DevOps Team Lead @Tufin

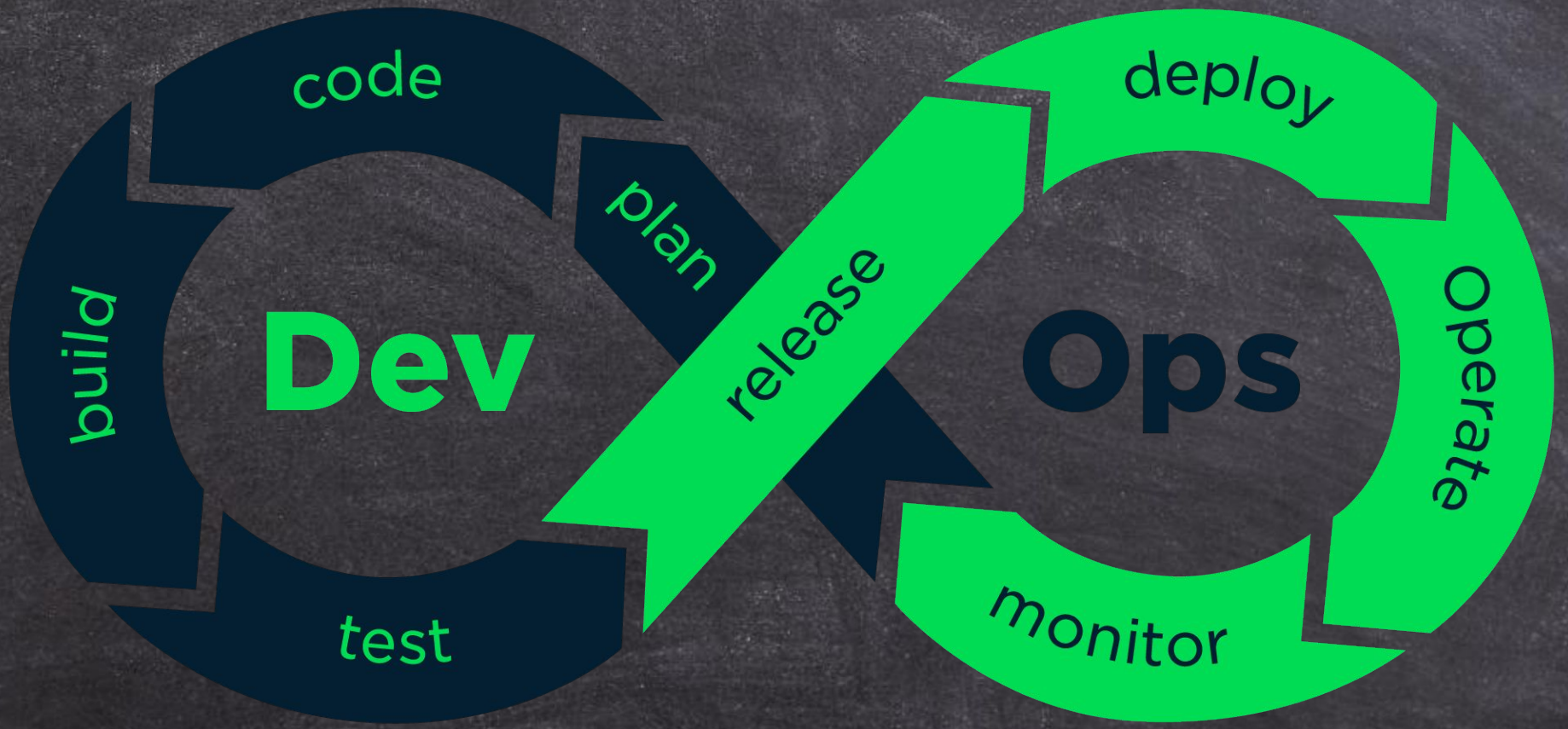
Previously on...





So What We will cover Today?

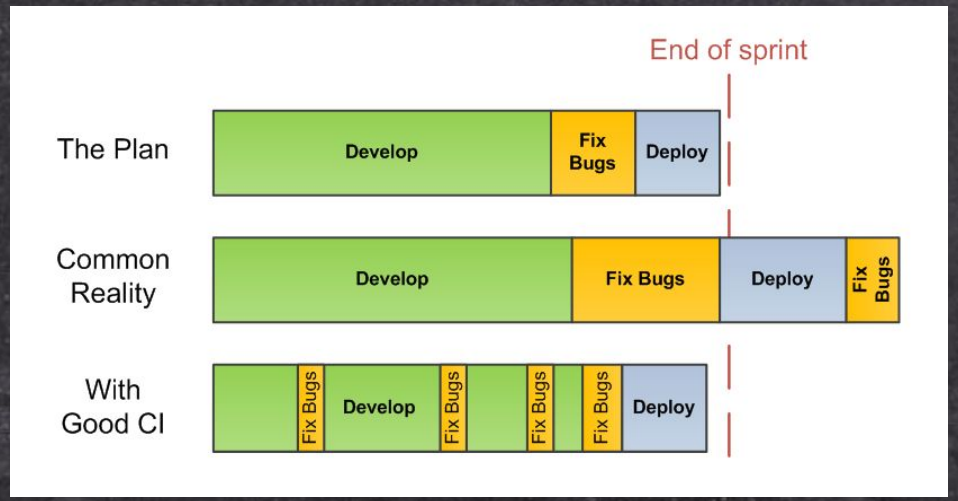
- What is CI and why do we need it?
- How do big hi-tech companies deploy applications?
- How to make a CD process smarter than anyone else?


code
plan
build
Dev
test
release
deploy
operate
Ops
monitor

So What is Continuous Integration ?

DevOps practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.



Continuous Integration Benefits

Improve Developer Productivity

Continuous integration helps your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs released to customers.

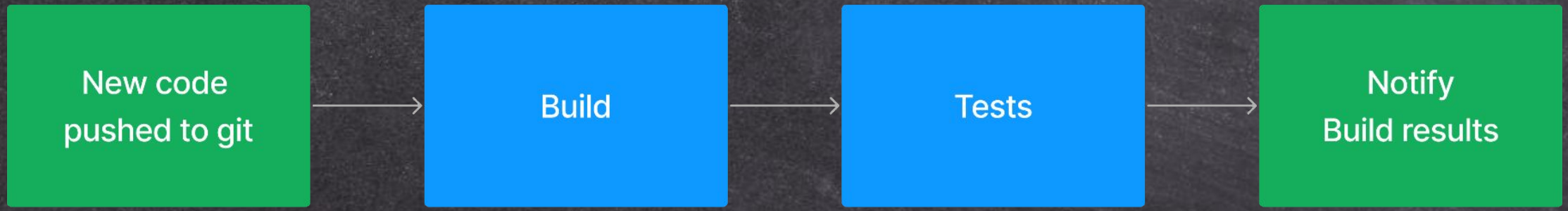
Find and Address Bugs Quicker

With more frequent testing, your team can discover and address bugs earlier before they grow into larger problems later.

Deliver Updates Faster

Continuous integration helps your team deliver updates to their customers faster and more frequently.

Source: AWS

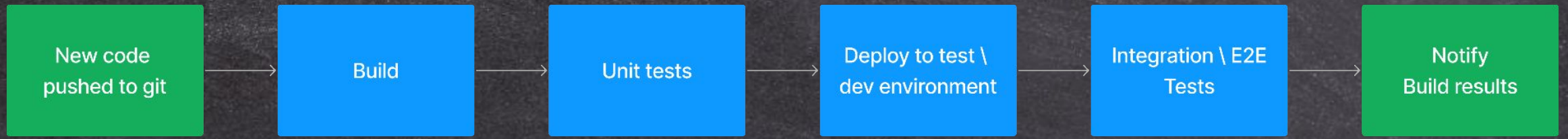

New code
pushed to git
Build
Tests
Notify
Build results

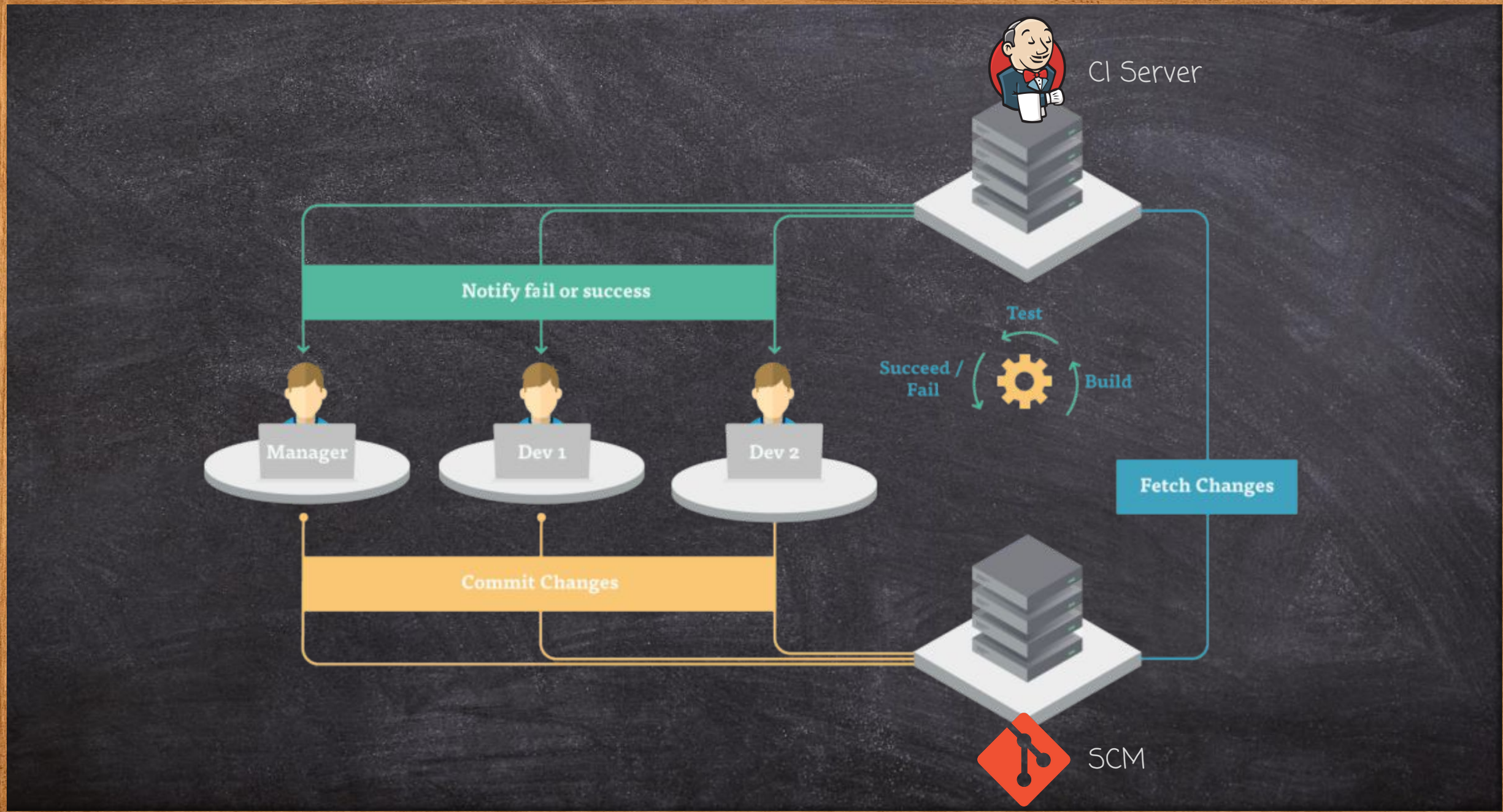
Softwar e Build ?!?

SOFTWARE TESTING

The different types of tests

- **Unit tests:** Very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software.
- **Integration tests:** Verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected.
- **End-to-end tests:** testing replicates a user behavior with the software in a complete application environment. It verifies that various user flows work as expected and can be as simple as loading a web page or logging in or much more complex scenarios verifying email notifications, online payments, etc...


New code pushed to git
Build
Unit tests
Deploy to test \ dev environment
Integration \ E2E Tests
Notify Build results


CI Server
Notify fail or success
Test
Succeed / Fail
Build
Manager
Dev 1
Dev 2
Fetch Changes
Commit Changes
SCM

Jenkins

Open source automation server which enables developers around the world to reliably build, test, and deploy their software

Continuous Integration and Continuous Delivery

As an extensible automation server, Jenkins can be used as a simple CI server or turned into the continuous delivery hub for any project.

Easy installation

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Linux, macOS and other Unix-like operating systems.

Easy configuration

Jenkins can be easily set up and configured via its web interface, which includes on-the-fly error checks and built-in help.

Plugins

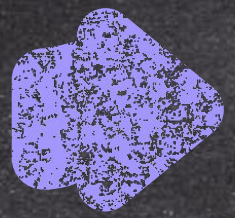
With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

Extensible

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

Distributed

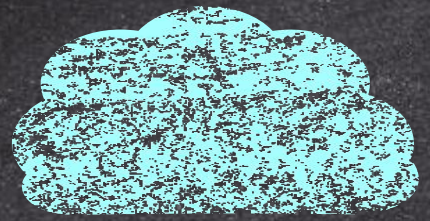
Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

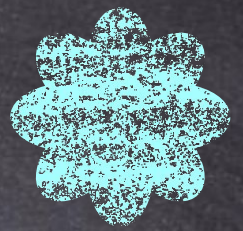


Jenkins & Docker

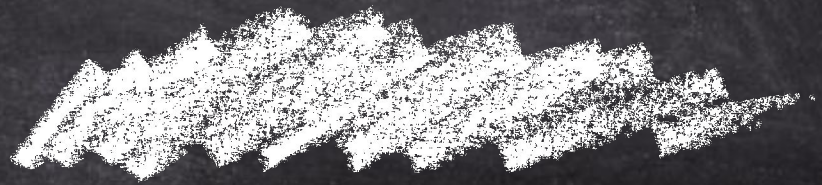
```
docker run -p 8080:8080 -p 50000:50000
        -v jenkins_home:/var/jenkins_home
    jenkins/jenkins:lts
```

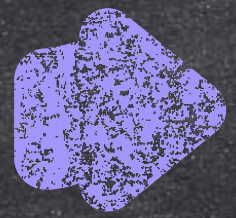
More info in: https://github.com/jenkinsci/docker/blob/master/README.md





YALLA SHOW US DEMO!!



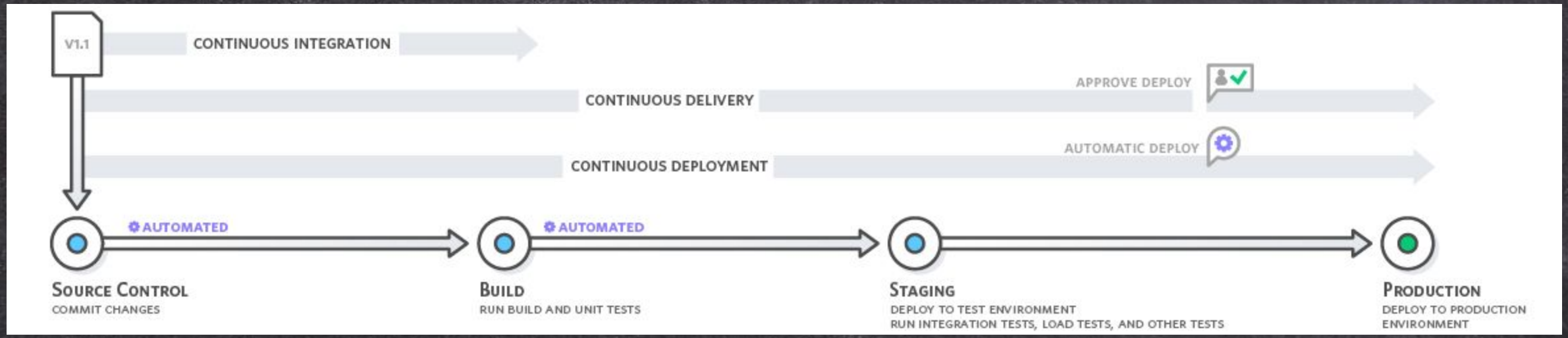


Continuous Delivery

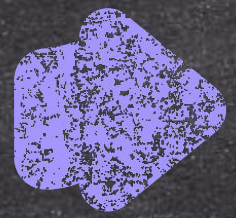
Practice where code changes are automatically prepared for a release to production.

A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage. When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Continuous Delivery vs. Continuous Deployment



Release?!?



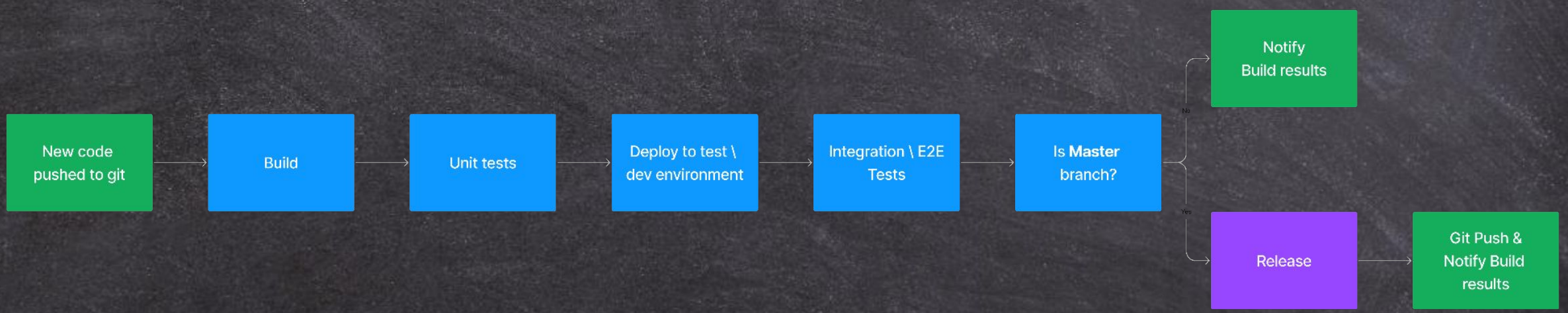
Auto Release Process

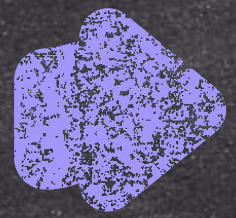
Branch != master/main (feature branch):

- CI Job (build, test, notify...)

Branch == master/main :

- CI Job (build, test)
- Release (bump version, tag & publish)
- CD Job (deploy to test/qa env -> production?)

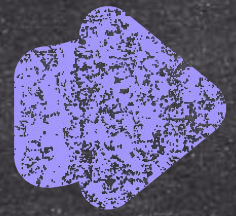

New code pushed to git
Build
Unit tests
Deploy to test \ dev environment
Integration \ E2E Tests
Is **Master** branch?
No
Notify Build results
Yes
Release
Git Push & Notify Build results



Jenkins Pipelines

Let you split the full process to a smaller steps with lots of benefits:

- Pipeline as a code
- Parallelism
- Steps can be paused & restarted
- Steps can wait for a human input before start



Jenkins Pipelines

Jenkins / mb-test / main / #1 / Pipeline Overview

#1

Rerun

main c7786ad Started 52 sec ago Queued 5.6 sec Took 6.9 sec Tests

Graph

Start Checkout SCM Build Test Release Post Actions End

Search

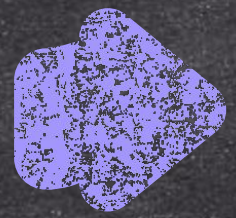
Checkout SCM 1.9s
Build 1.1s
Test 0.62s
Release 0.26s
Post Actions 22ms

Build 1.1s Started 1m 8s ago Jenkins

Going to install project dependencies 8ms

npm install 1.0s

```
0 + npm install
1 npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do not use
  it. Check out lru-cache if you want a good and tested way to coalesce async requests by a key
  value, which is much more comprehensive and powerful.
2 npm warn deprecated rimraf@3.0.2: Rimraf versions prior to v4 are no longer supported
3 npm warn deprecated abab@2.0.6: Use your platform's native atob() and btoa() methods instead
4 npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
5 npm warn deprecated domexception@2.0.1: Use your platform's native DOMException instead
6 npm warn deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and
  performance.timeOrigin.
```



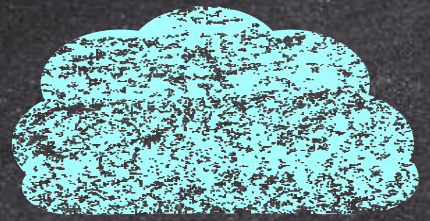
Jenkins Pipelines

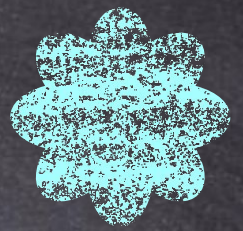
```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //
            }
        }
        stage('Test') {
            steps {
                //
            }
        }
        stage('Deploy') {
            steps {
                //
            }
        }
    }
}
```

Jenkinsfile is a text file that contains the definition of a Jenkins Pipeline and is checked into source control.

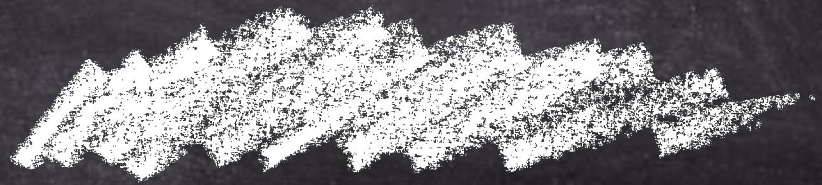
Consider the following Pipeline which implements a basic three-stage continuous delivery pipeline

Pipeline syntax: https://www.jenkins.io/doc/book/pipeline/syntax/





YALLA SHOW US DEMO!!



Deployment Strategies

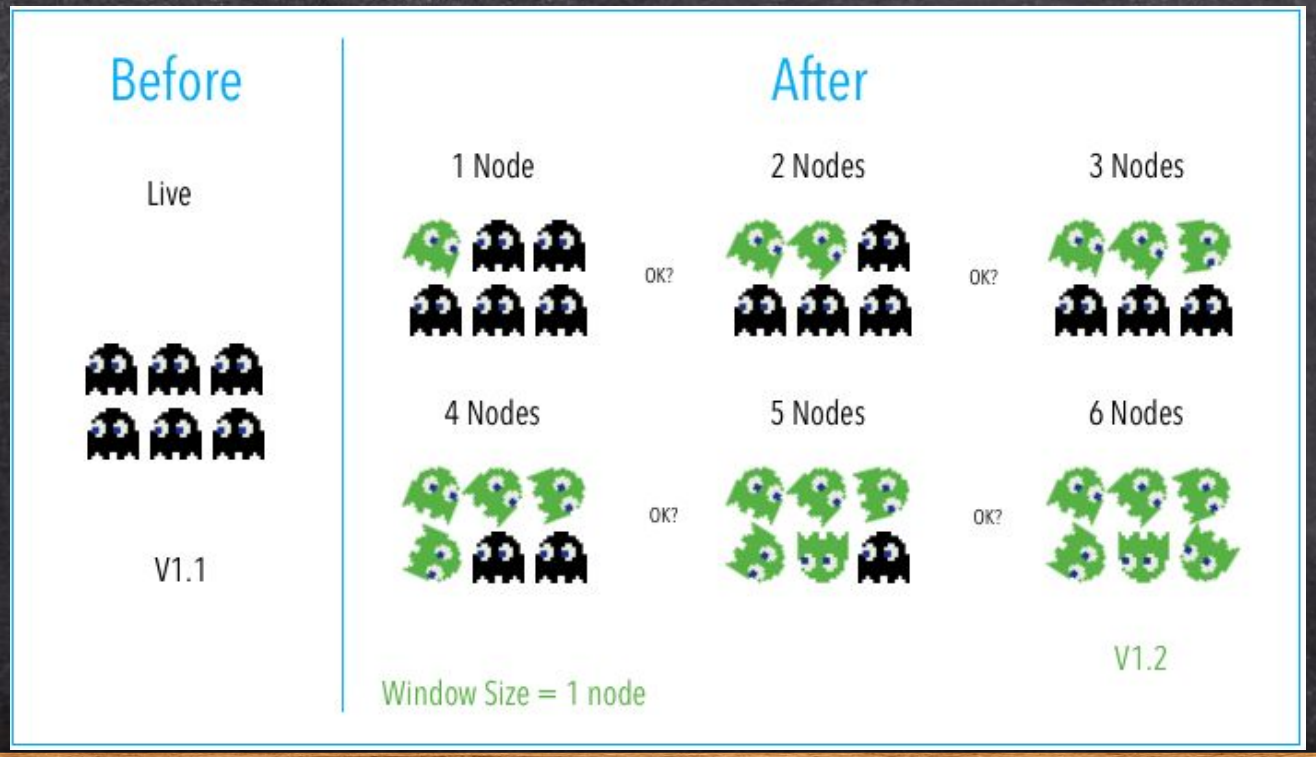
The Basic Deployment

In a basic deployment, all nodes within a target environment are updated at the same time with a new service or artifact version.

Deployment Strategies

Rolling Deployment

All nodes in a target environment are incrementally updated with the service or artifact version in integer N batches.



Deployment Strategies

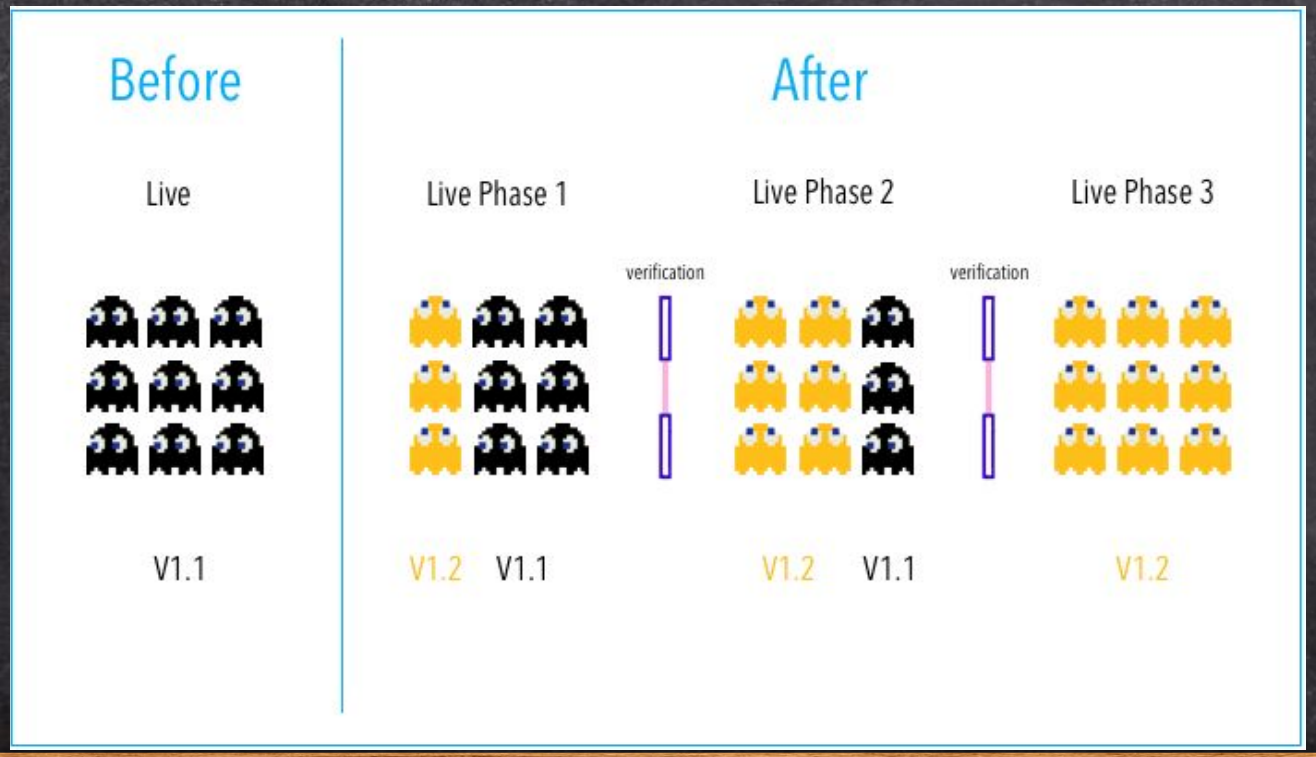
Blue-Green Deployment

User traffic is shifted from the green environment to the blue environment once new changes have been testing and accepted within the blue environment.

Deployment Strategies

Canary Deployment

Releases an application or service incrementally to a subset of users. All infrastructure in a target environment is updated in small phases



CI \ CD Competition

Jenkins

Circle CI

TeamCity

Github Actions

Gitlab CI

Jfrog Artifactory

(And 342,876 more...)

Let's summarize...

We learned What is CI and why do we need it?

We stated to play with Jenkins, build our first freestyle and pipeline jobs

We learned How do big hi-tech companies deploy their applications?

Thank you!

Do you have any questions?

Dor Alteresku

@doralteres