**Content**

[1. Concepts 2](#_Toc13053160)

[1. Continuous Integration 2](#_Toc13053161)

[2. Jenkins 6](#_Toc13053162)

[1. Installation 6](#_Toc13053163)

[2. Master and Slave Architecture 6](#_Toc13053164)

[3. Terminology 6](#_Toc13053165)

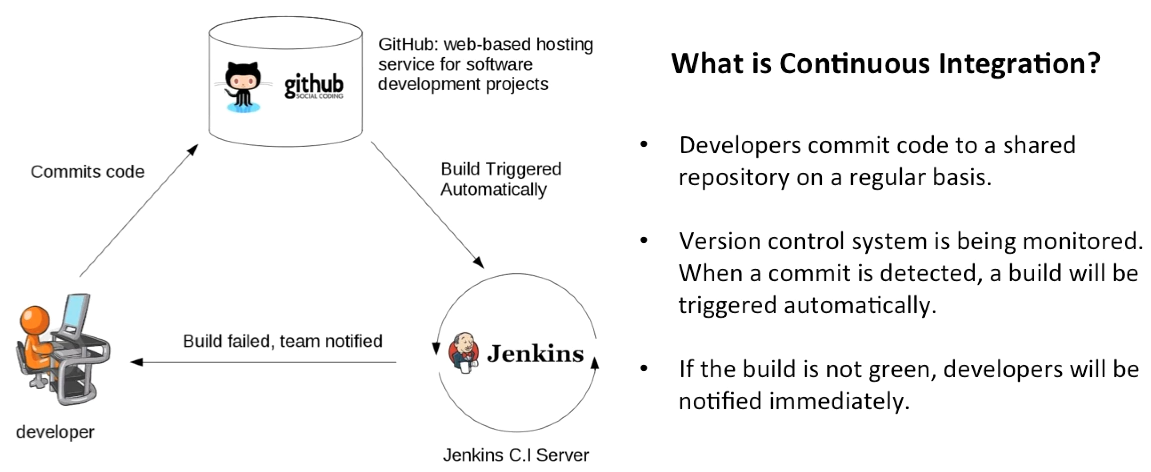
[3. Automating your Delivery Pipeline from GitHub to Amazon EC2 instance using Jenkins 8](#_Toc13053166)

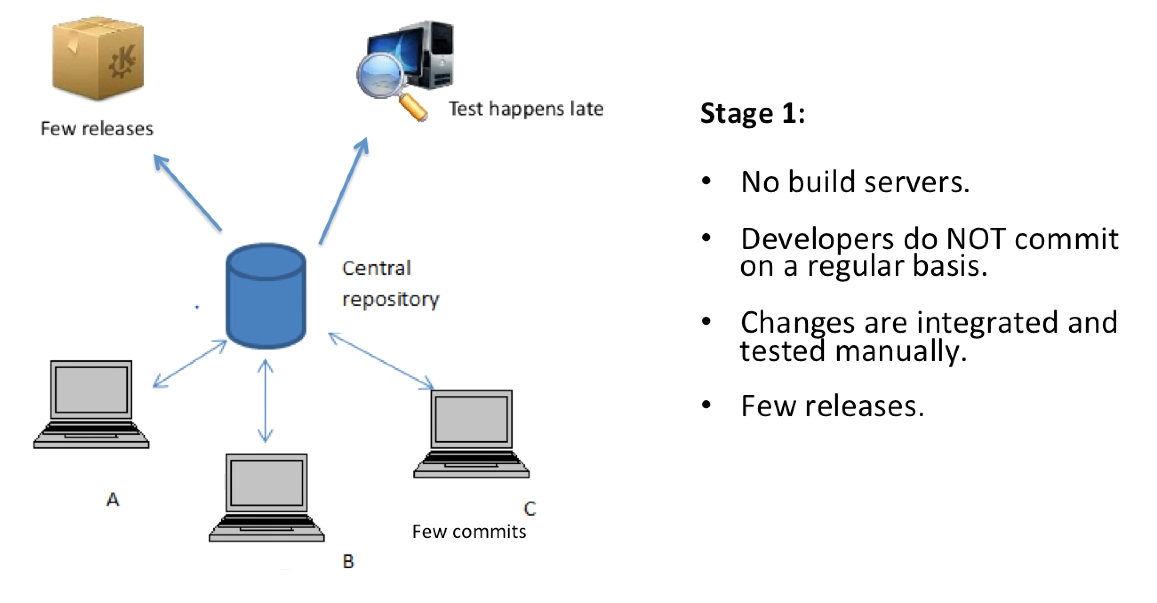
[4. AWS CodePipeline Integration With CodeDeploy and GitHub 10](#_Toc13053167)

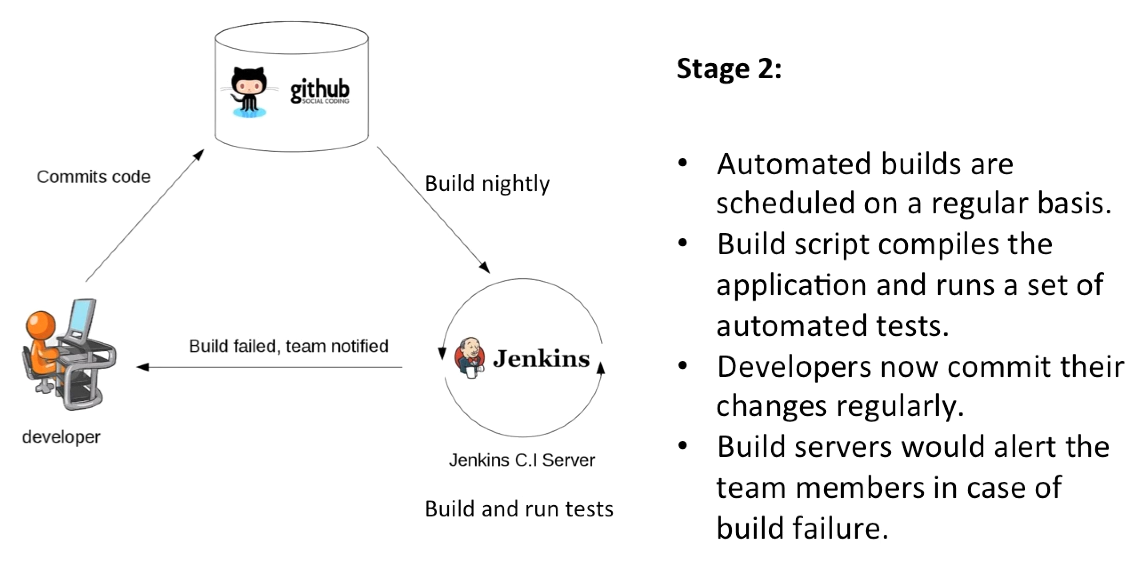
[1. AWS CodePipeline 10](#_Toc13053168)

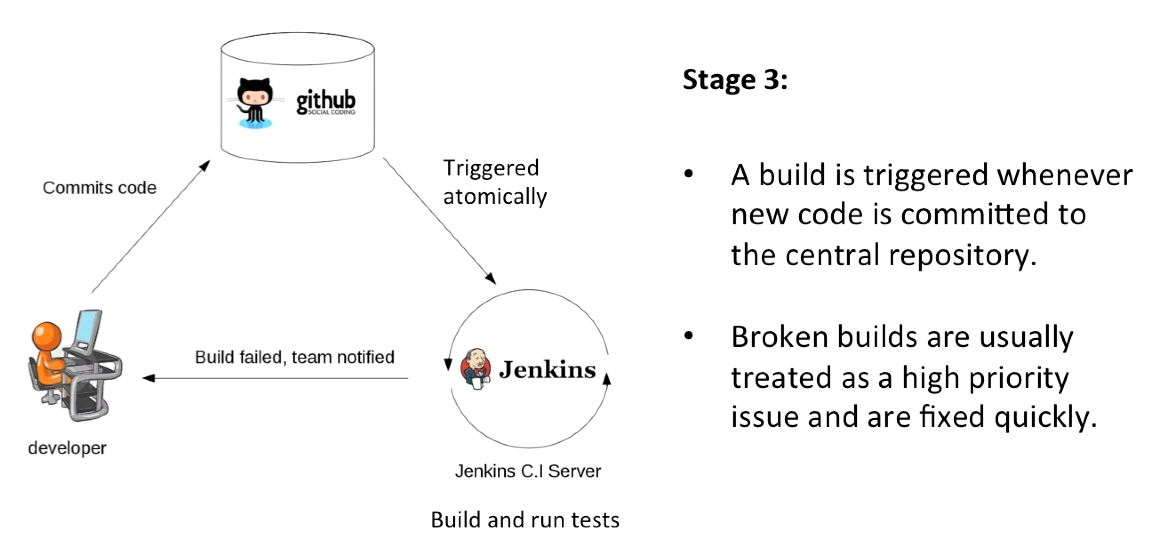
[2. GIT – Webhooks 10](#_Toc13053169)

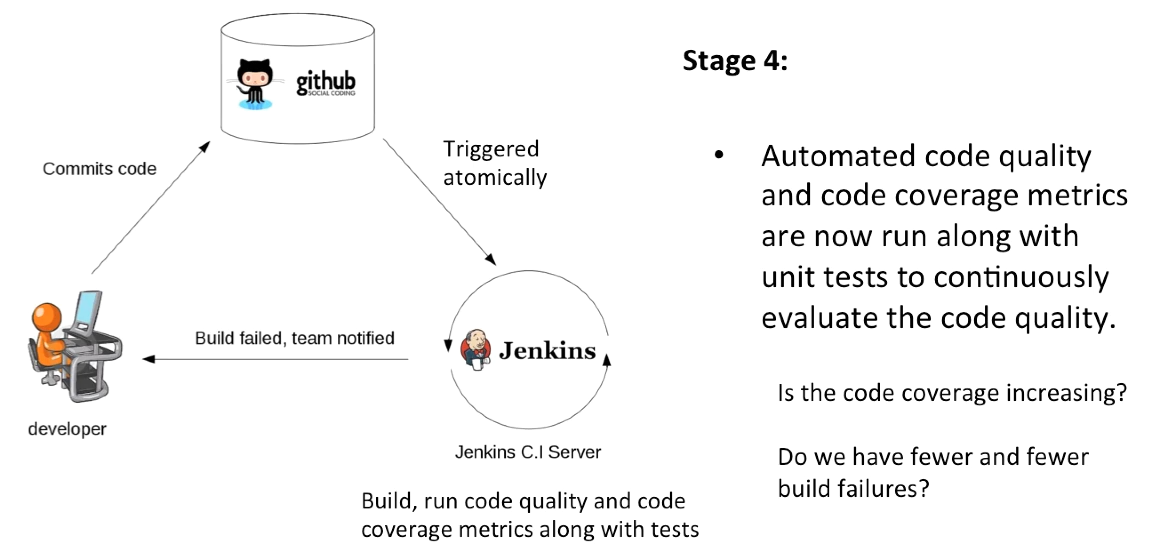
1. Concepts
   1. Continuous Integration

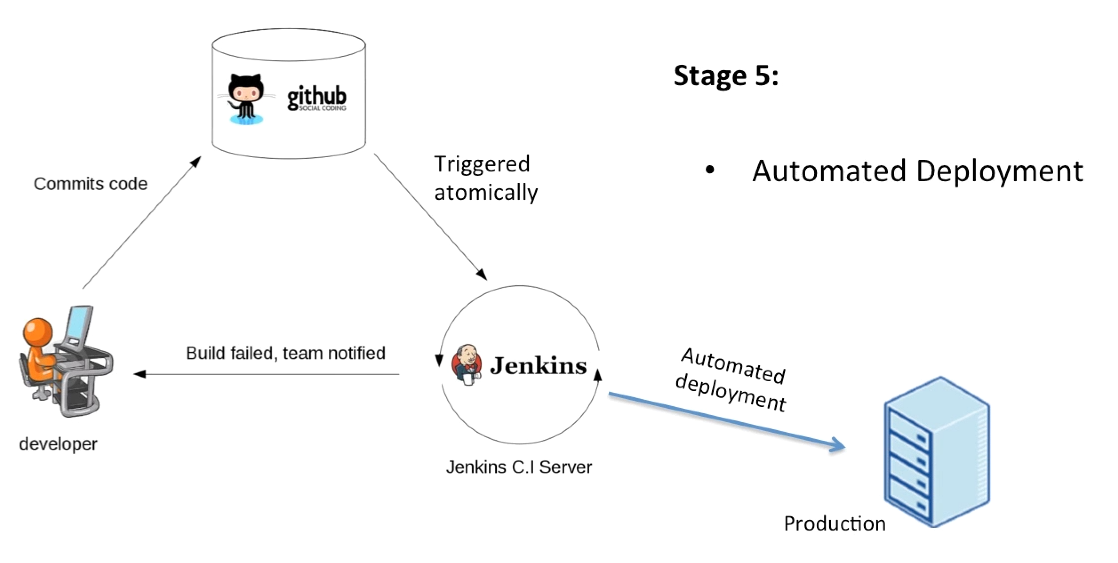












**Continuous Integration**

The practice of merging development work with the main branch constantly (developers integrate their work frequently to the same location, such as master branch).

**Continuous Delivery**

Software development methodology, where the release process is automated. Continual delivery of code to an environment once the code is ready to ship. This could be staging or production. The idea is the product is delivered to a user base, which can be a QUs or customers for review and inspection (test). Although every successful software change can be immediately released to production with CD, not all changes need to be released right away.

**Continuous Integration** is focused on **automatically building and testing code**, as compared to **Continuous Delivery**, which **automates** **the** entire **software release process** up to production.

**Continuous Deployment**

The deployment or release of code to production as soon as it is ready.

Non-hosted solutions: Jenkins

Hosted solutions: circleci

Others:

* Buddy
* TeamCity
* TravisCI
* Bamboo
* GitLab CI
* Codeship
* codefresh

1. Jenkins

* CI and build server
* Used to manually, periodically, or automatically build software development projects
* It is an open source CI tool written in Java
* Language support : Groovy, PHP, .NET, Ruby, C/C++, Android, Java, Python
* Easy to use, lots of plugins
  1. Installation

JAVA\_HOME env var points to the installation path for the JDK

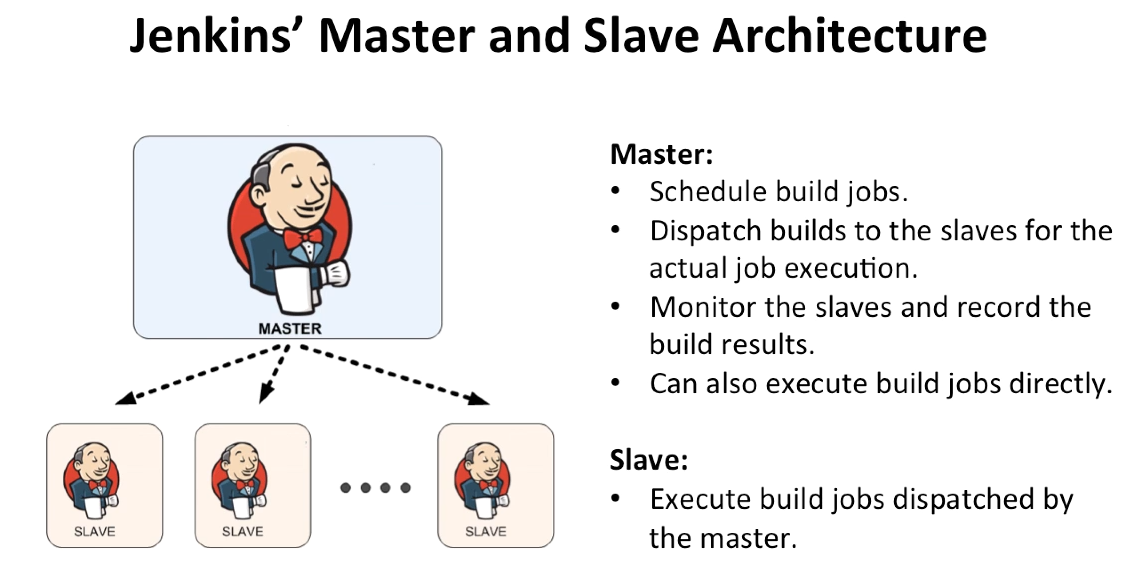
Address: <http://localhost:8080>

echo %JAVA\_HOME%

cd %JENKINS\_HOME%

jenkins [stop | start]

* 1. Master and Slave Architecture



Slave is a small java program that listens to the request

* 1. Terminology

**Job / Project**

Those 2 terms are used interchangeably. They all refer to runnable tasks that are controlled / monitored by Jenkins.

**Slave / Node**

Slaves are computers that are set u to build projects for a master. Jenkins runs a separate program called “slave agent” on slaves. When slaves are registered to a master, a master starts distributing loads to slaves. Node is used to refer to all machines that are part of Jenkins grid, slaves and master.

**Executor**

A separate stream of builds to be run on a node in parallel. A node can have 1 or more executors.

**Build**

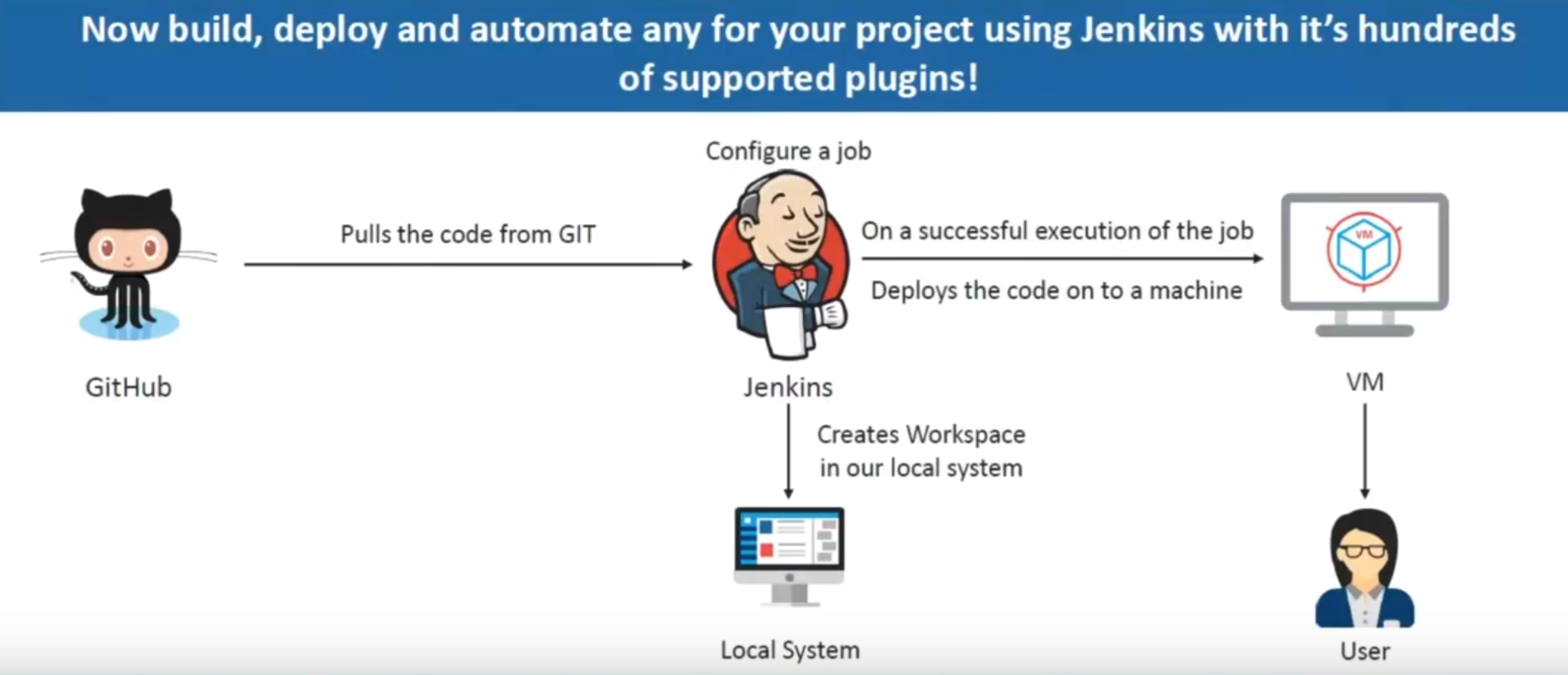
Is a result of one of the projects.

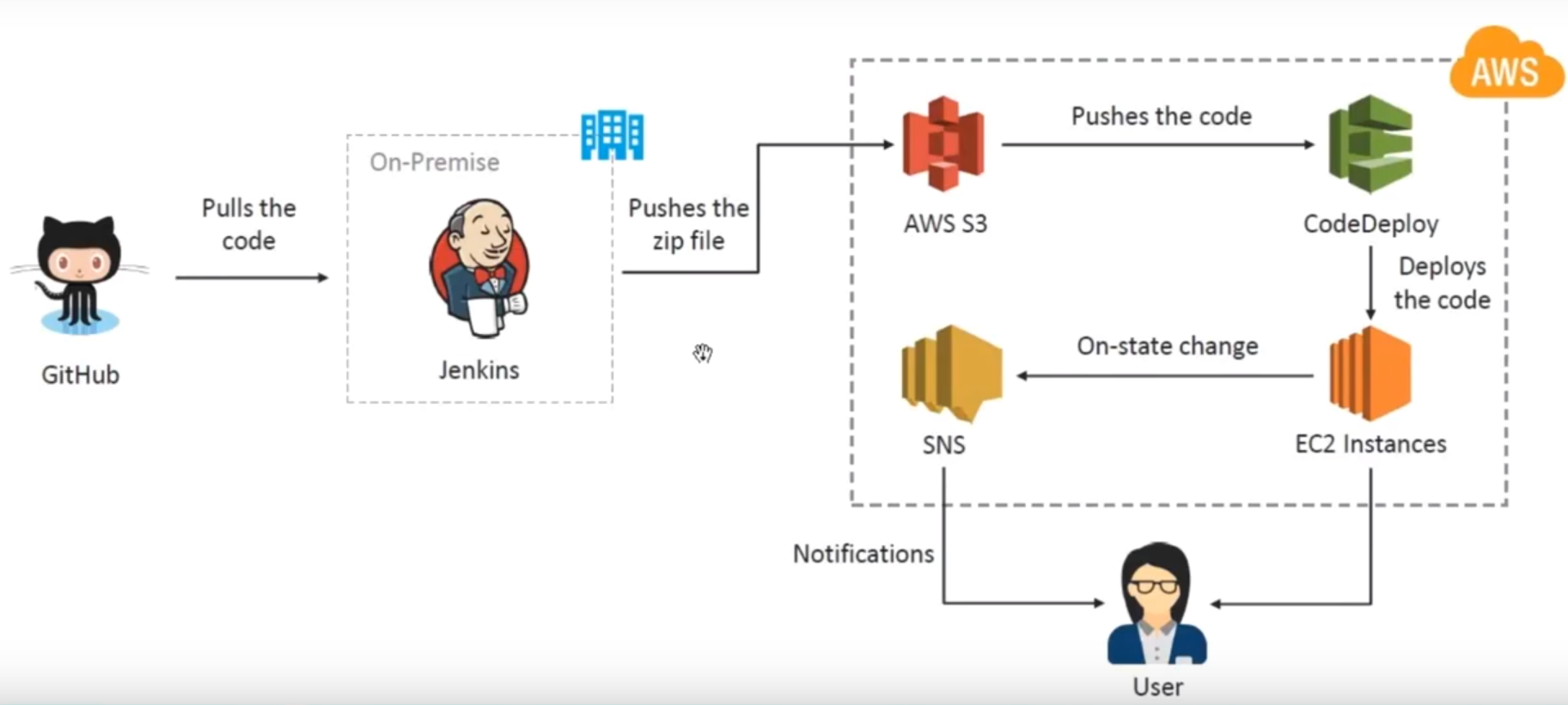
**Plugin**

A Plugin, like plugins on any other system, is a piece of software that extends the core functionality of the core Jenkins server.

1. Automating your Delivery Pipeline from GitHub to Amazon EC2 instance using Jenkins

Jenkins AWS plugin???

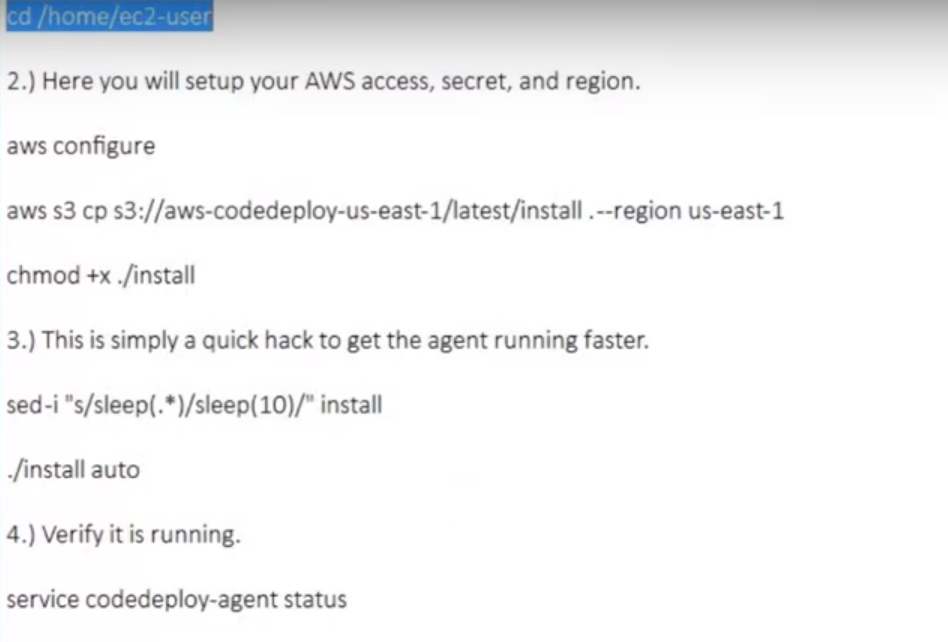




* Jenkins is on my local machine

Steps:

1. Ec2 needs to communicate with S3 (create role)
2. Install code deploy agent on Ec2 (code deploy agent???))

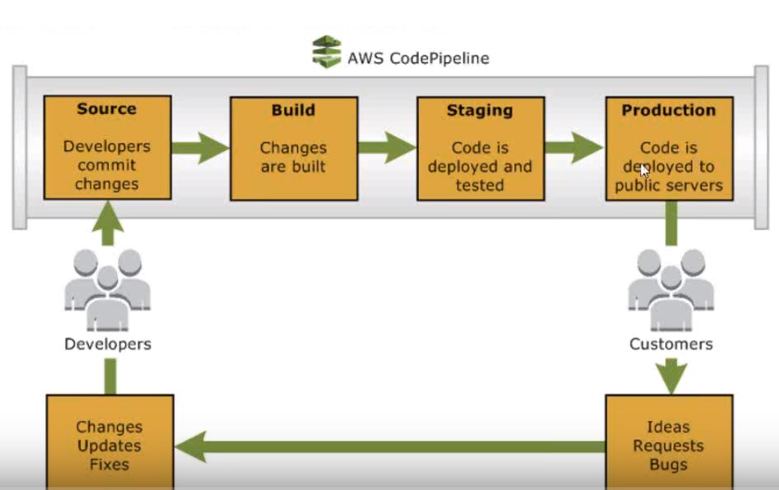


1. Code Deploy:
   * Custom deployment
   * Crate application (app name, deployment group name)
   * Choose Auto Scaling group or Ec2 instances (in this we proceed w Ec2 instances)
   * Deployment configuration (CodeDeployDefault . OneAtTime – for this tutorial)
   * + Code Deploy needs a role (god knows how, ec2 is needed for sure)
2. Create Deployment
   * Go to Deployment groups -> Actions -> Deploy new revision
   * Configure your git repo (github, not gitlab w mfa )
   * Content options – override the content
3. Jenkins
   * Pust-build Action – where it is specified to deploy the application onto the AWS code deploy.
   * Region, bucket, access key, secret access key … and aws shits

More info: <https://www.youtube.com/watch?v=LFkGtg-ZTko>

1. AWS CodePipeline Integration With CodeDeploy and GitHub
   1. AWS CodePipeline

AWS CodePipeline is a continuous delivery service you can use to model, visualize, and automate the steps required to release your software. You can quickly model and configure the different stages of a software release process. AWS CodePipeline automates the steps required to release your software changes continuously.



* 1. GIT – Webhooks

Webhooks notify a remote service by issuing an HTTP POST when a commit is pushed to the repository. AWS Lambda receives the HTTP POST through Amazon API Gateway, and then downloads a copy of the repository. It places a zipped copy of the repository into a versioned S3 bucket. AWS CodePipeline can then use the zip file in S3 as a source; the pipeline will be triggered whenever the Git repository is updated.

**Webhooks** allow you to build or set up integrations, such as GitHub Apps or OAuth Apps, which subscribe to certain events on GitHub.com. When one of those events is triggered, we'll send a HTTP POST payload to the webhook's configured URL. Webhooks can be used to update an external issue tracker, trigger CI builds, update a backup mirror, or even deploy to your production server. You're only limited by your imagination.

When configuring a webhook, you can choose which events you would like to receive payloads for.

Each event type has a specific payload format with the relevant event information.

