https://devops.com/creating-continuous-integration-delivery-pipeline/

What is DevOps? ✔

DevOps type of agile relationship between 'development' and 'IT operations'.

Goal behind DevOps is to change & improve the relationship between these two business units by 'better communication' and 'collaboration'

Providing a platform for Developers and software testers improve the efficiency, quality and speed to market within the software development world.

Theme behind DevOps is automation, 1 person doing a 4 man job by automating a process. The concept behind this approach is to save time and cost for an organisation.

Continuous Integration ✔

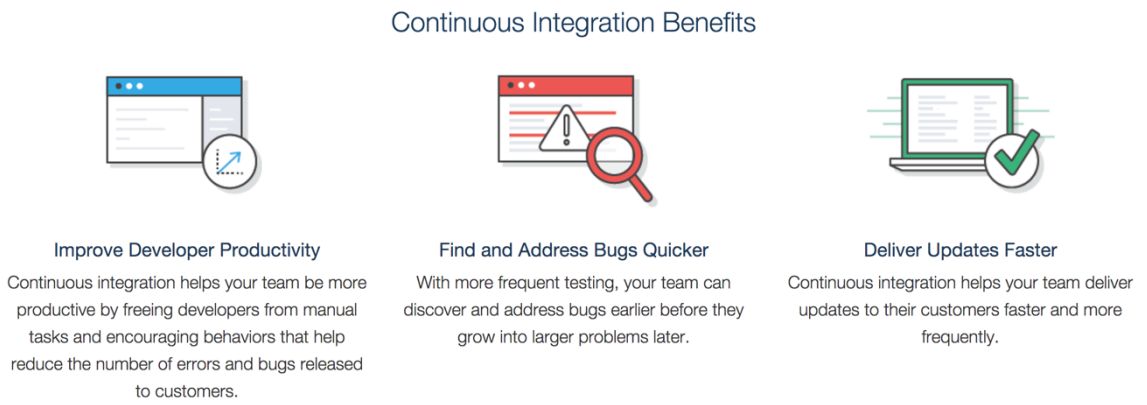
'Continuous Integration', 'Continuous Deployment', and 'Continuous Delivery' are key approach behind modern development teams. It enables a team to safely 'build', 'test', and 'deploy' their code.

This modern development practice helps teams release quality code faster and continuously.

Advantages:

Transforming software operations to optimize for efficiency, quality, and reliability

https://dzone.com/articles/9-bene-ts-of-continuous-integration



HOME OFFICE - Louy Heidar - 14:30pm

Technical skills in DevOps, and your soft skills related to communication

Tech learnt at the Academy (i.e. Jenkins, Git, **AWS**, Docker, Ansible).

AWS in particular

drawing out and explaining a CI pipeline

additional tools you can add (Jira, Maven, JUnit, etc.).

Docker

Advantages of using Docker?

Rapid application deployment – containers include the minimal runtime requirements of the application, reducing their size and allowing them to be deployed quickly.

Portability across machines
This container can be transferred to another machine that runs Docker, and executed there without compatibility issues.

Simplified maintenance – Docker reduces effort and risk of problems with application dependencies.

Ansible

Master & Agent Configuration

This is based on a master agent configuration.

Master sending an instruction to the Slave or the agent machines ✔

Slave carries out that command ✔

The advantages of using this approach

Jenkins slave – if you are not building very simple applications, you probably have several machines attached to your Jenkins (or other CI system) instance and use them to execute many builds simultaneously. ✔

This approach allows more efficiency on large scale projects to save time and cost ✔

Verify that Ansible has been installed

Confirmation - Ansible installed on vitual machine) via Terminal (antiMasterDG)
Ansible –version

Successful Establish connection between for Master Agent configuration.

Add Agent machine to hosts group Creating 'hosts' group

Navigating to correct file directory
/etc/ansible/hosts.

Add IP of Agent machine using Nano file editor

Test for successful connection for master agent configuration

Test from Master that connection was successful
#Command executed from Master VM
$ ansible all -i hosts -u vagrant -m setup

Success message should appear in green, which indicates that have been not problems and a successful connection has been established.

Successful Establish connection between for Master Agent configuration.

Showing successful connection to host
ansible all –m ping

Ansible Playbooks

Playbooks are Ansible's configuration, deployment, and orchestration language. They describe a policy you want your remote systems to enforce as a set of steps in a general IT process.

Advantages of using Ansible playbooks is that with every playbook execution, only the applied changes are executed for each of the defined tasks. This optimised approach saves time particularly when provisioning instances within a virtual environment ✔

Ansible Playbook Syntax

Playbooks are expressed in YAML format. Normally a model of a 'configuration' or a 'process'. ✔

Ansible Syntax Note/ Comment

YAMl is a human-readable data serialization language ✔

Whitespace indentation is used to denote structure.

Have to be careful with the syntax and formatting to reduce time troubleshooting, syntax realted issues. ✔

Ideal apprach for deploying applications

Jenkins Installed on Agent VM:

Apporach 1:

Downloading package from remote server using apt-get and installing from remote location.
Disadvantage:

If this applied to many machines it would slow down the automation process since it would need to be downloaded Jenkins remotely each time to install on every machine without using the local install file (jenkins_2.1_all.deb).
This approach would not be practical and efficient with more than one machines

```
- name: creating configlink
  command: "{{ item }}"
  with_items:
- "apt-get install -y -f"
- "apt-get install -y jenkins"
- "service jenkins start"
```

Apporach 2:

Using locally stored package to deploy Jenkins on default port (80:80). Idempotent approach used to prevent the same process being repeated multiple times i.e. in this case packages

will not be installed twice if the package already exists, only the updated changes are processed.

```
- name: install_jenkins
  apt: deb="/tmp/shared/jenkins_2.1_all.deb" state=present force=yes

- name: run_jenkins
  service: name=jenkins state=started enabled=yes
```

Jira Installed on Agent VM:

File Name: response.varfile

varfile for mapping the configuration settings for Jira during Installing.
Port set to run on 8091

Git

Git (/ɡɪt/[7]) is a version control system for tracking file changes and coordinating work on files among teams. It is primarily used for source code management in software development,[

GitHub: Central Repository to push all code when working within a team, has the ability to track updates and changes, particularly when code breaks.

Push Code to New repository

1. Create a new repository on GitHub. To avoid errors, do not initialize the new repository with *README*, license, or `gitignore` files. You can add these files after your project has been pushed to GitHub.
2. Open Terminal.
3. Change the current working directory to your local project.
4. Initialize the local directory as a Git repository.

```
5. git init
```

6. Add the files in your new local repository. This stages them for the first commit.

```
7. git add .
8. # Adds the files in the local repository and stages them for commit. To unstage a file, use 'git reset HEAD YOUR-FILE'.
```

9. Commit the files that you've staged in your local repository.

```
10.         git commit -m "First commit"
11.         # Commits the tracked changes and prepares them to be pushed to a remote repository. To remove this commit and modify the file, use 'git reset --soft HEAD~1' and commit and add the file again.
```

12. At the top of your GitHub repository's Quick Setup page, click to copy the remote repository URL.
13. In Terminal, add the URL for the remote repository where your local repository will be pushed.

```
git remote add origin remote repository URL
# Sets the new remote
git remote -v
# Verifies the new remote URL
```

18. Push the changes in your local repository to GitHub.

```
git push -u origin master
# Pushes the changes in your local repository up to the remote repository you specified as the origin
```

What is Cloud Computing and the Advantages of AWS

It's a way of enabling companies to consume shared computing, storage and other resources faster and more efficiently compared 'building' and 'operating' their own IT infrastructure. ✔

- **Cost-Effective** – Consume only the amount of compute, storage and other IT resources needed. No long-term commitment, minimum spend or up-front investment is required.

- **Elastic and Scalable** – Quickly add and subtract resources to applications to meet customer demand and manage costs.

- **Security** highest and strictest security best practices. AWS conducts regular and thorough audits to demonstrate the security of its infrastructure.

- **Flexible** – Improve overall productivity and time to market without the need for IT to learn new skills.

AWS Questions and Answers

I have some private servers on my premises, also I have distributed some of my workload on the public cloud, what is this architecture called?

Explanation: This type of architecture would be a hybrid cloud. Why? Because we are using both, the public cloud, and your on premises servers i.e the private cloud. To make this hybrid architecture easy to use, wouldn't it be better if your private and public cloud were all on the same network(virtually). This is established by including your public cloud servers in a virtual private cloud, and connecting this virtual cloud with your on premise servers using a VPN(Virtual Private Network).

Amazon EC2 security groups?

A Security group is just like a firewall, it controls the traffic in and out of your instance. In AWS terms, the inbound and outbound traffic. The command mentioned is pretty straight forward, it says create security group, and does the same. Moving along, once your security group is created, you can add different rules in it. For example, you have an RDS instance, to access it, you have to add the public IP

address of the machine from which you want access the instance in its security group.

5. How is stopping and terminating an instance different from each other?

Starting, stopping and terminating are the three states in an EC2 instance, let's discuss them in detail:

- **Stopping and Starting** an instance: When an instance is stopped, the instance performs a normal shutdown and then transitions to a stopped state. All of its Amazon EBS volumes remain attached, and you can start the instance again at a later time. You are not charged for additional instance hours while the instance is in a stopped state.
- **Terminating** an instance: When an instance is terminated, the instance performs a normal shutdown, then the attached Amazon EBS volumes are deleted unless the volume's*deleteOnTermination* attribute is set to false. The instance itself is also deleted, and you can't start the instance again at a later time.

7. When will you incur costs with an Elastic IP address (EIP)?

An **Elastic IP** address is a static IPv4 address designed for dynamic cloud computing.

Explanation: You are not charged, if only one Elastic IP address is attached with your running instance. But you do get charged in the following conditions:

- When you use more than one Elastic IPs with your instance.
- When your Elastic IP is attached to a stopped instance.
- When your Elastic IP is not attached to any instance.

16. What are the best practices for Security in Amazon EC2?

There are several best practices to secure Amazon EC2. A few of them are given below:

- Use AWS Identity and Access Management (IAM) to control access to your AWS resources.
- Restrict access by only allowing trusted hosts or networks to access ports on your instance.
- Review the rules in your security groups regularly, and ensure that you apply the principle of least
- Privilege – only open up permissions that you require.
- Disable password-based logins for instances launched from your AMI. Passwords can be found or cracked, and are a security risk.
- **19. Can S3 be used with EC2 instances, if yes, how?**

- Yes, it can be used for instances with root devices backed by local instance storage. By using Amazon S3, developers have access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. In order to execute systems in the Amazon EC2 environment, developers use the tools provided to load their Amazon Machine Images (AMIs) into Amazon S3 and to move them between Amazon S3 and Amazon EC2.

Jenkins

Difference between 'Job' and 'Build'

Job is a process that is defined e.g. compiling, testing or packaging applications
Artifacts are produce as result of process that are defined

Job can have many builds associated with it
Build represent the results of executing a job

1) Mention what is Jenkins?

Jenkins is an open source tool with plugin built for continuous integration purpose. The principle functionality of Jenkins is to keep a track of version control system and to initiate and monitor a build system if changes occur. It monitors the whole process and provides reports and notifications to alert.

CI Pipeline

Start off with
Source Code Management – Git a version control system and Central Repository to push all code when working within a team,

Continuous integration server - Jenkins that pulls and builds the application

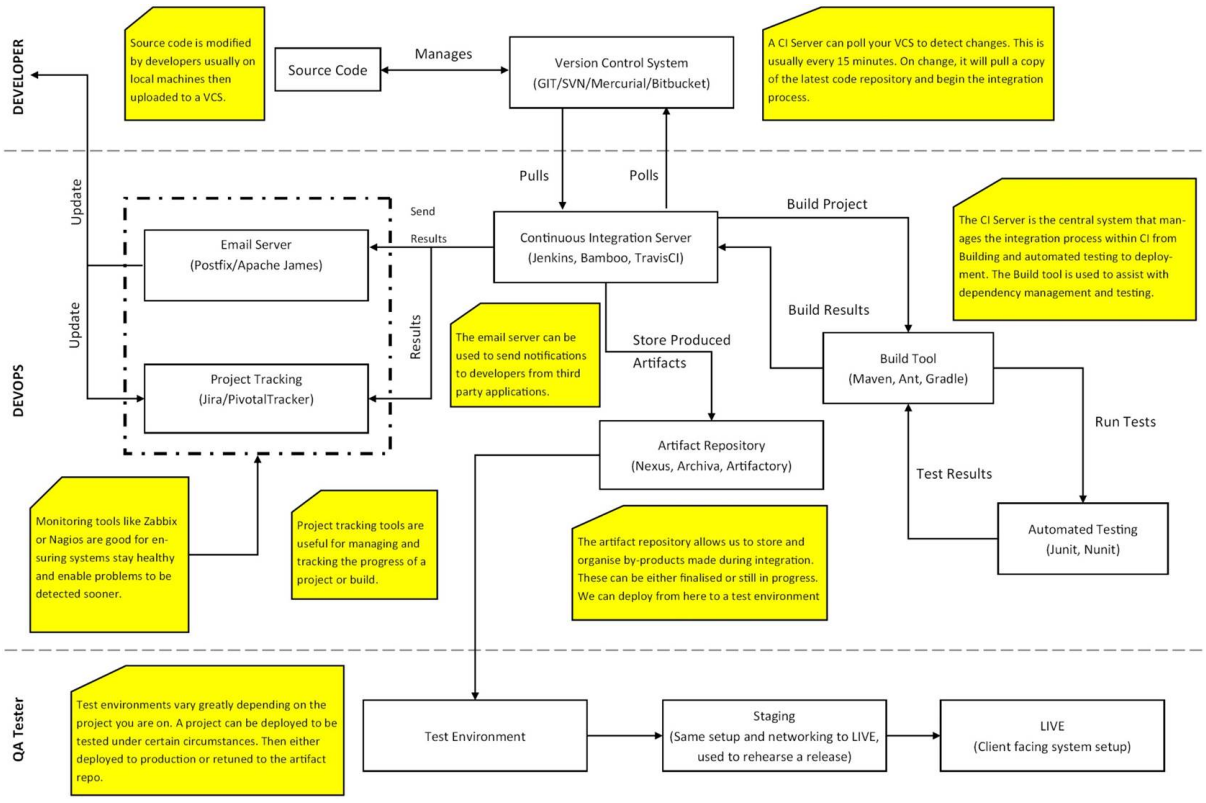
Build Tools: Maven - Build tools such as Maven would be used to build the project and validate project builds

Artifact repository - Nexus - Use Nexus with your CI server to deploy successfully built snapshots and releases which can be available to other developers. Advantages of using an Artifact repository is it can be version controlled.

Application would then be deployed to a test server for testing with the use of **testing frameworks** such as - JUnit

Deployment to production – where you would use **configuration management** and deployment tools such as Ansible to complete the deployment on the production server for release.

After the product its is releases its is then **monitored** by **tools** like Zabbix and Splunk that provide feedback for e.g. Downtime



The delivery pipeline can be broken down into a few major buckets of work, or stages, as mentioned below.

In my opinion—and, again, trying to keep things simple for now—the stages can be broken down as follows:

1. Source Code Control (Management)
2. Build Automation
3. Unit Test Automation (could also include Integration Testing here as well)
4. Deployment Automation
5. Monitoring – not included in this discussion, and can be added at any time

https://devops.com/creating-continuous-integration-delivery-pipeline/