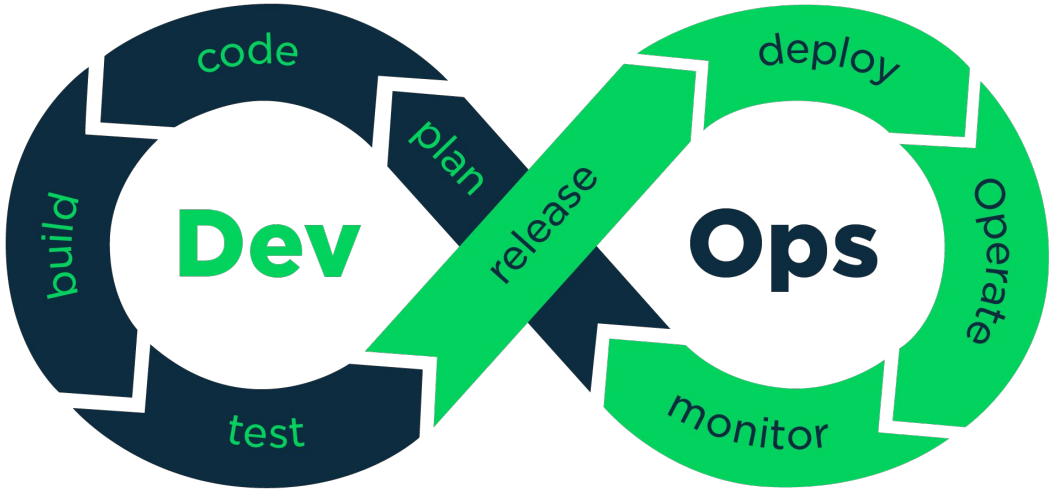

code
plan
build
Dev
test
release
deploy
Ops
Operate
monitor

DevOps

ARROUMA Mohamed
DevOps consultant & Scrum Master at Sofrecom
(Orange Labs Tunisia)

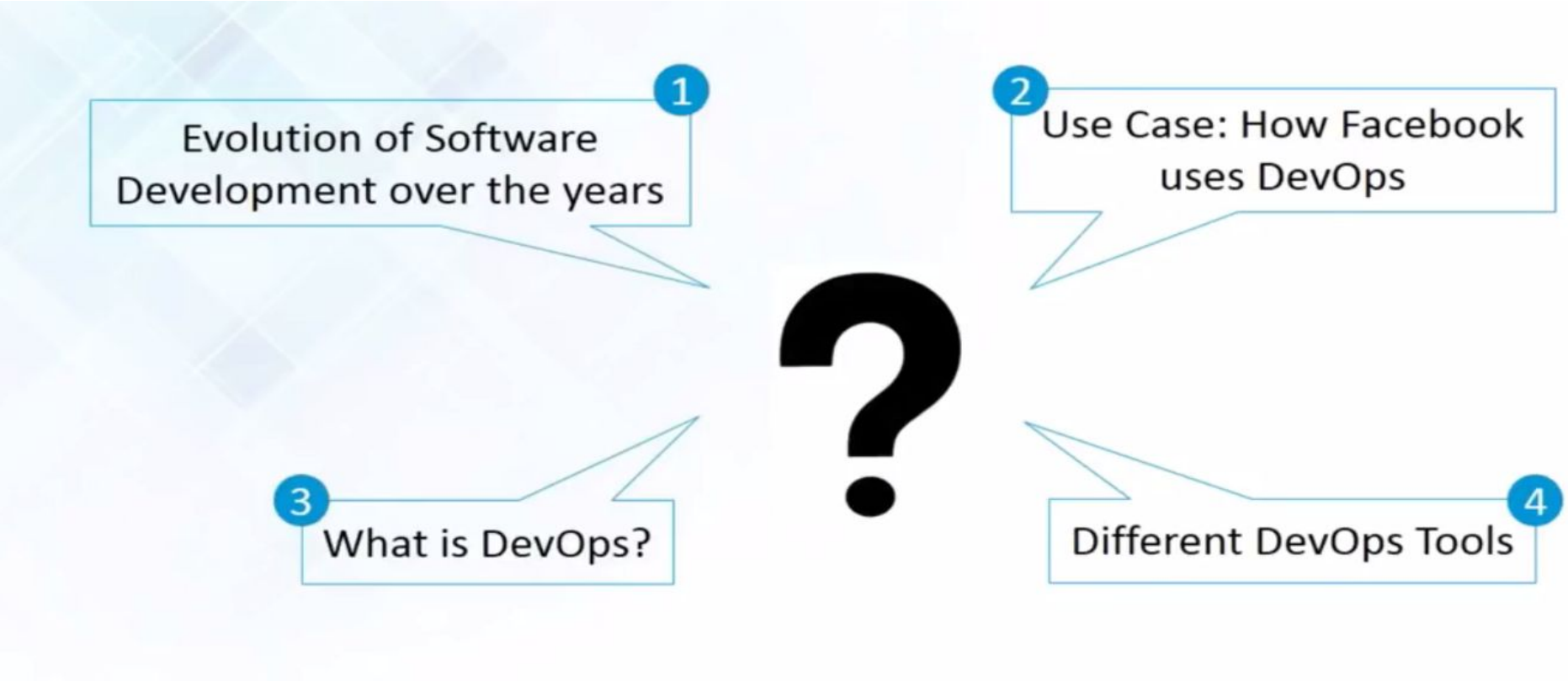
Sofrecom
The Know-How Network

- ✔ RedHat Certified System Administrator (RHCSA)
- ✔ RedHat Certified System Engineer (RHCE)
- ✔ RedHat Certified Specialist in OpenShift Administration
- ✔ RedHat Certified Specialist in Ansible Automation
- ✔ Scaled Agile Framework Certified ®SAFe4 Agilist

?

DEV

OPS



Do you know the following :

Traditional Waterfall Model

Best suited when:

- Complete Requirements are clear and fixed
- Product definition is stable

Agile Development

Best suited when:

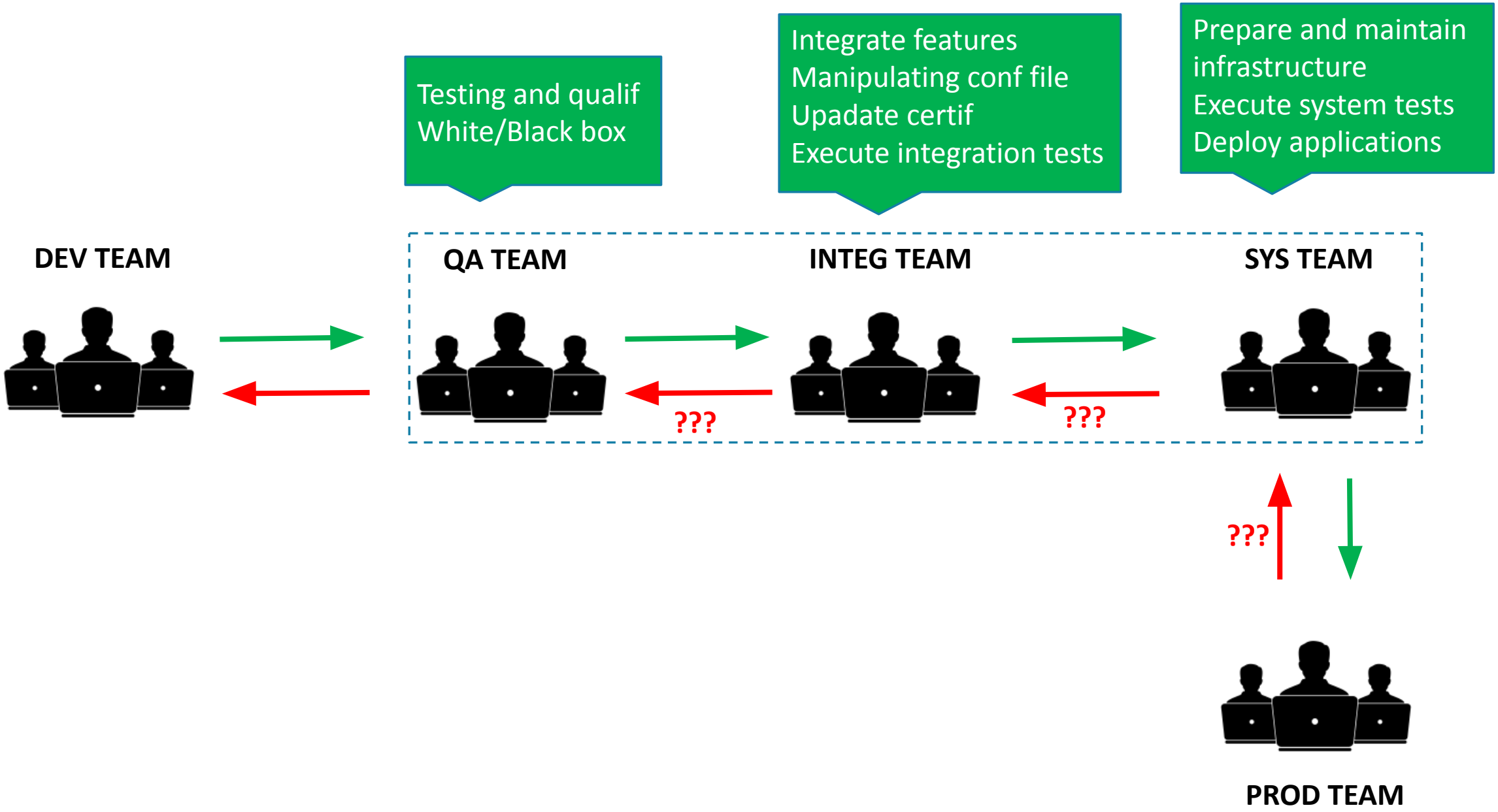
- Requirements change frequently
- Development needs to be fast

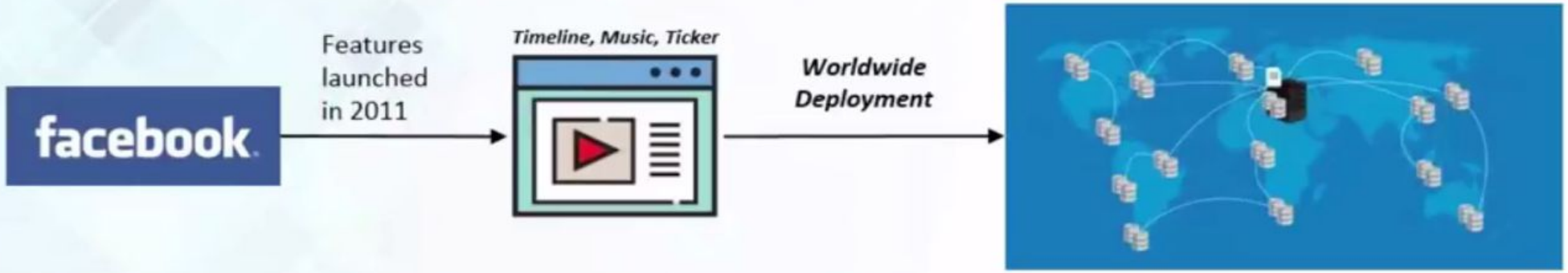
DevOps Approach

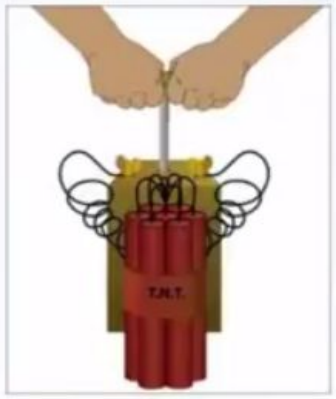
Best suited when:

- Requirements change frequently
- **Development** needs to be Agile
- **Operations** needs to be Agile

Development


Testing and qualif
White/Black box
Integrate features
Manipulating conf file
Upadate certif
Execute integration tests
Prepare and maintain infrastructure
Execute system tests
Deploy applications
DEV TEAM
QA TEAM
INTEG TEAM
SYS TEAM
???
???
???
PROD TEAM





Challenges they faced that day

- Features released to 500 million users → Heavy Website traffic → Server Meltdown
- Mixed responses from users which lead to no conclusion

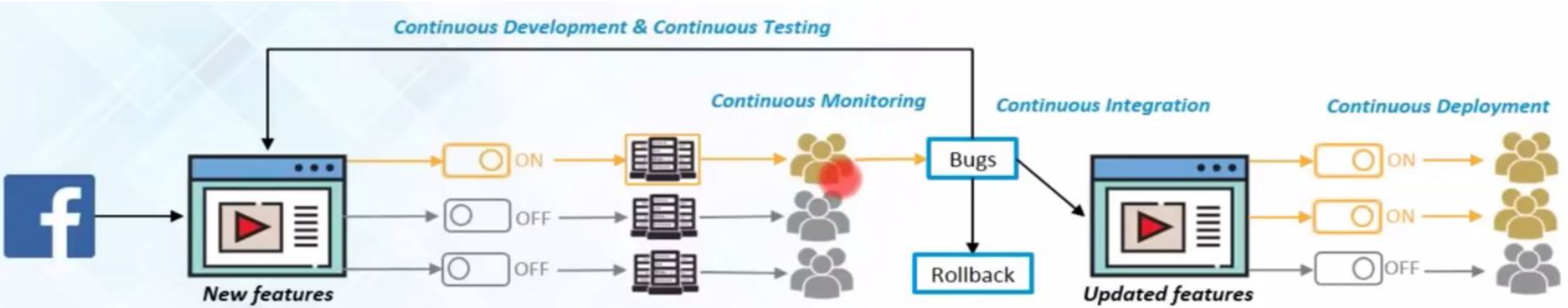
Use Case 2011 : FaceBook

FaceBook Comes Up with Dark Launching

According to Dark Launching Technique:

- The new features are first deployed on a smaller & specific user base.
- They are continuously monitored and the feedbacks are continuously developed and tested.
- Once the features are stable, they are deployed on other user bases in multiple releases.

The Dark Launching Technique

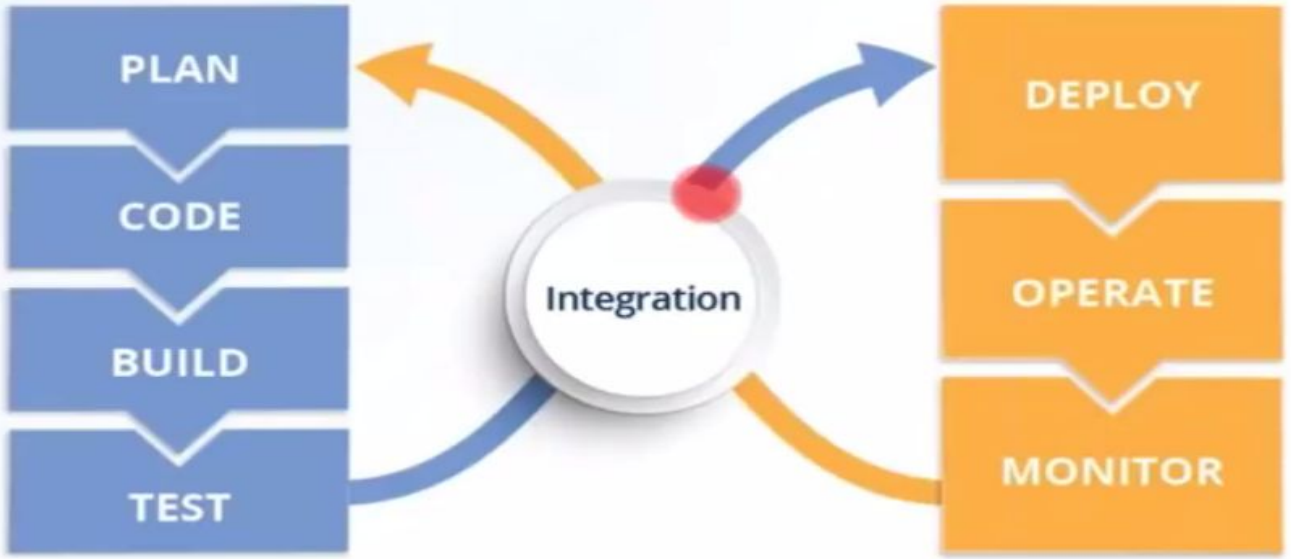


To Implement Dark launching, the below activities are fundamental as they lie at the heart of the DevOps lifecycle:

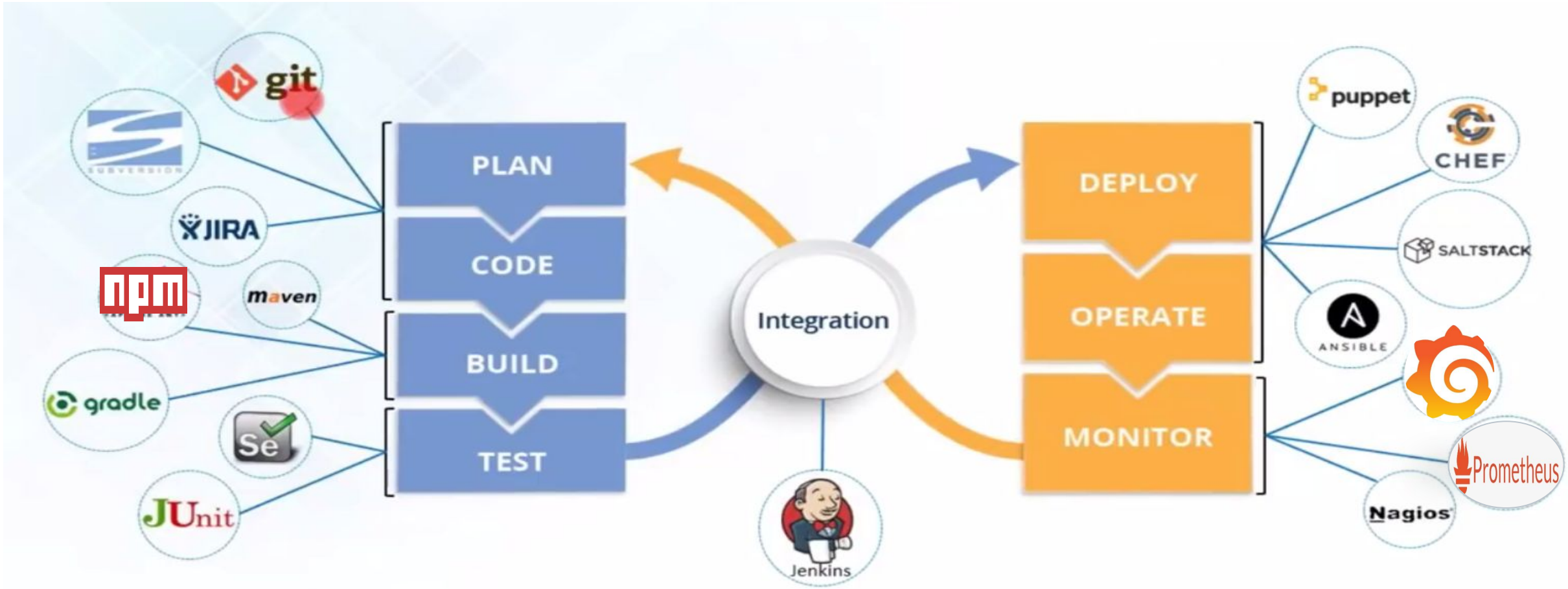
- Continuous Development
- Continuous Testing
- Continuous Integration
- Continuous Deployment
- Continuous Monitoring

The Dark Launching Technique

DevOps is a Software Development approach which involves Continuous Development, Continuous Testing, Continuous Integration, Continuous Deployment and Continuous Monitoring of the software throughout its development lifecycle



What is DevOps ?



DevOps Tools

Continuous Development

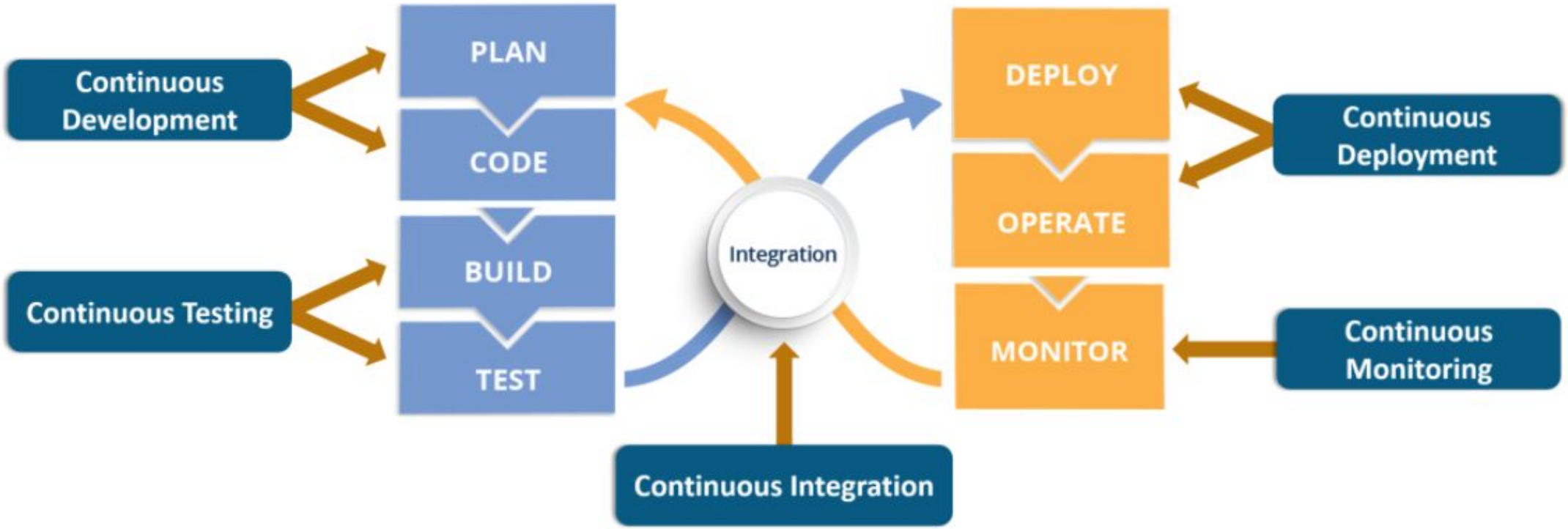
Continuous Testing

Continuous Integration

Continuous Deployment

Continuous Monitoring

DevOps Lifecycle Phases



DevOps LifeCycle Phase

Continuous Development

- This phase involves ‘**planning**‘ and ‘**coding**‘ of the software application’s functionality.
- The vision of the project is decided during the ‘planning’ phase.
- Code can be done in Any Language but maintain by **Version Control System**.

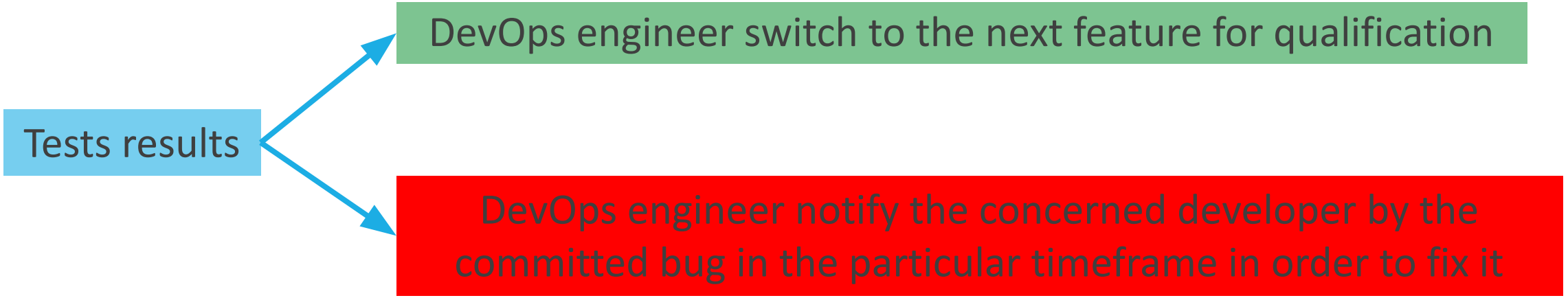
Why Code Versioning is Important

- Versions are maintained to hold a single source of Application.
- Using Centralized single source code, **Operations can access the same code what they plan to release**.
- Easy to **Rollout** the faulty snippet of code or complete release.

Continuous Testing

- Is the second phase of the product life cycle
- This phase involves '**build**' and '**test**' of the software application's functionality.
- Compiling all source code files, generating missing modules, install the dependencies after any source code modification or particular time duration
- Checking application status on specific branch after each build
- Executing automated verification/validation tests on every build by the integration server (Jenkins)

Continuous Testing

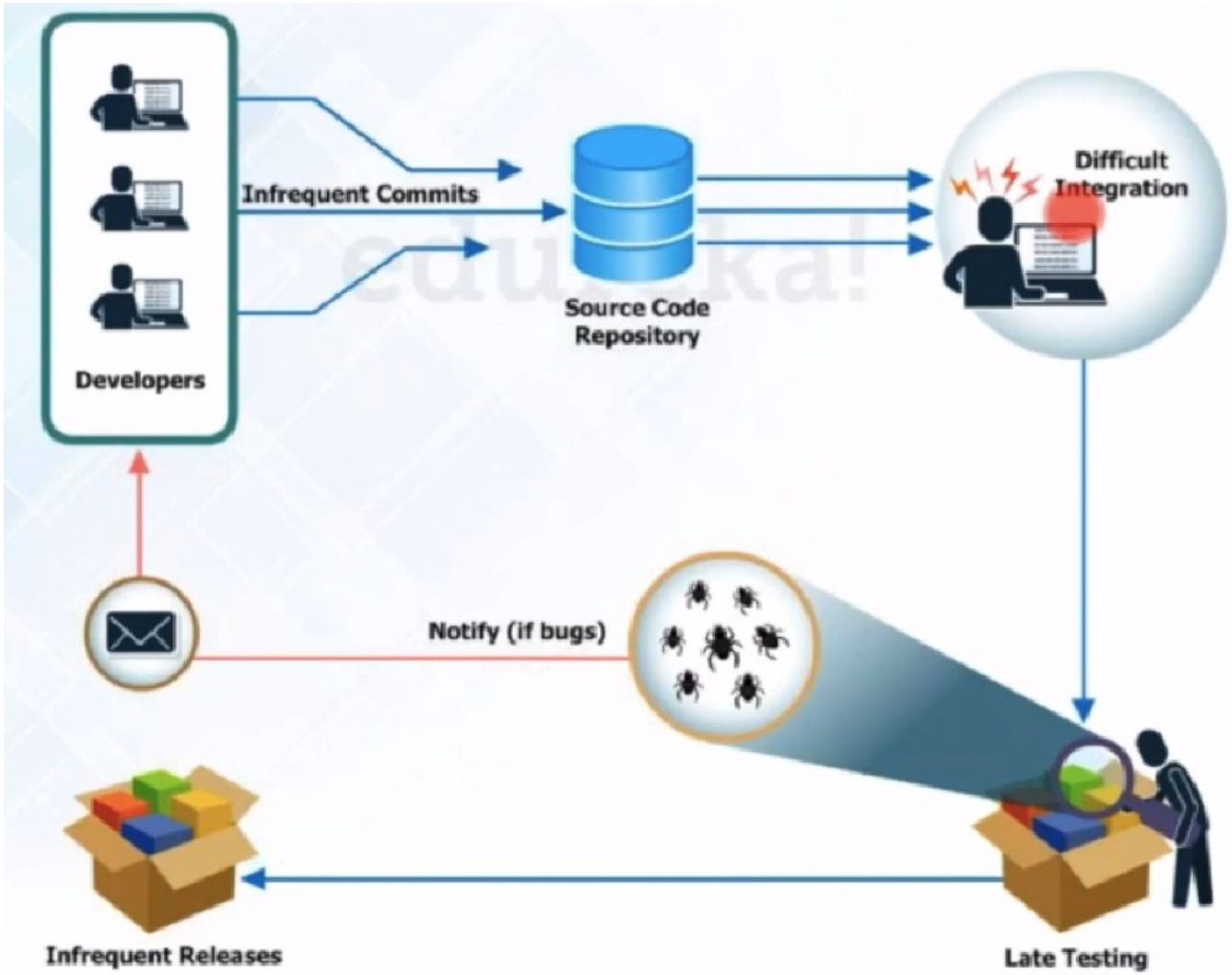


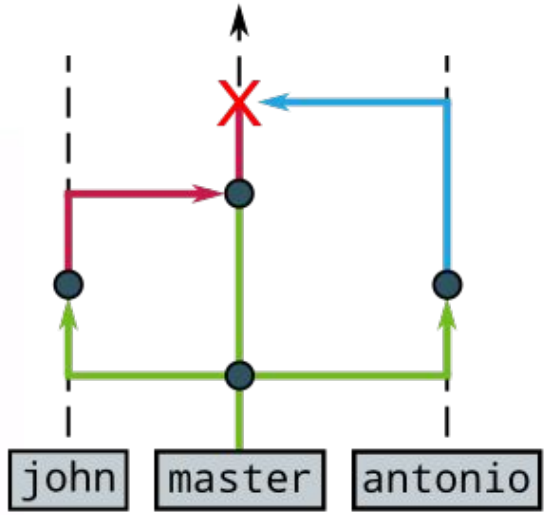
👍 We can easily find out the bugs !!

Continuous Integration

- Is the continuous merging of the developed features in the main source code of the application after making sure that there is no regression on the build and the test process.

Use Case : Manual Integration





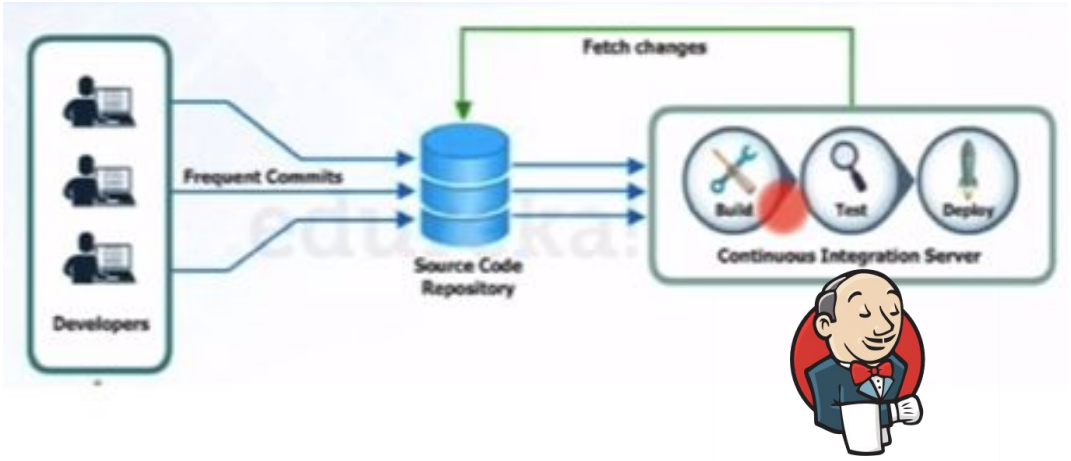
4) Antonio tried to merge, but Git accused a file **conflict**

3) John merged his branch into the master branch

2) John and Antonio edited the same file in parallel

1) There was a file on a Git repository

Continuous Integration



Scheduled/triggered jobs will automatically detect conflict quickly and locate them easily instead of discovering issues in the end of feature development

- Continuous Integration brings multiple benefits to your organisation:
- Say goodbye to long and tense integrations
- Increase visibility enabling greater communication
- Catch issues early
- Spend less time debugging and more time adding features
- Stop waiting to find out if your code's going to work
- Reduce integration problems allowing you to deliver software more rapidly

Continuous Deployment

Consist of automating the operations of preparing all the prerequisites for an application to run:

- configuring servers
- Networking administration,
- Controlling connections,
- Packages installation
- Launching build process
- Executing system tests and integration tests)

Continuous Deployment benefits

- Gain a lot of time and reduce complexity by automating repetitive tasks
- Improve Operational Confidence
- Provide a single view across all applications and environments.
- Improve overall productivity.
- Enhances teamwork by focusing more on work that deliver business value

Continuous Monitoring

Is the regular check of servers health by implementing several tools and running performance tests in order to detect risk issues.

- Monitoring of infrastructure performance (Metrics)

- Monitoring the application stability (log rotate, transactions, latency)


kibana


Grafana


Nagios


Prometheus

Course Summary

- Git
- Jenkins
- Ansible
- Docker
- Kubernetes
- AWS






GIT : Version Control System

GIT & GITHub : Version Control System

- Why GIT?
- Understanding the Core Concept of GIT
- Features of GIT
- Master in GIT Commands
- Compare GIT Commands and Working Model
- How GIT play vital Role in DevOps
- Exposure to GITHub
- Bonus Materials

VCS

- **Version Control System** - VCS helps a software team manage changes to source code over time.
- Version control system keeps track of every modification of the code in a special kind of database.
- VCS helps team to rollback to previous version in case of any issue with specific Version.
- **The need of Version Control System is primordial !!!**

VCS

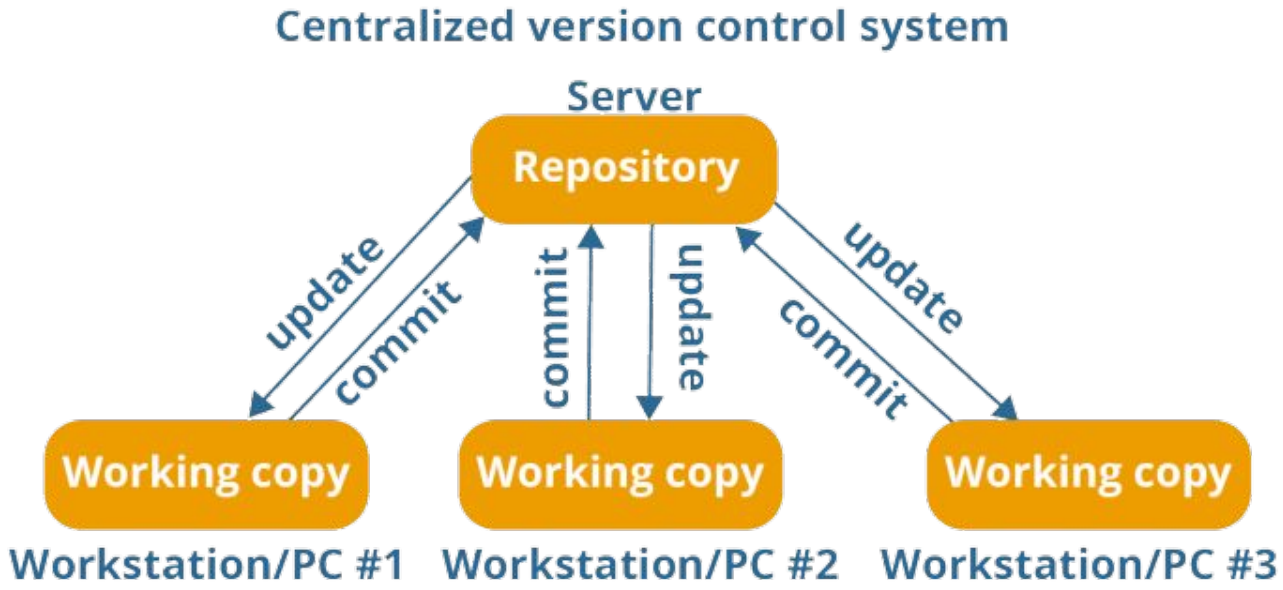
Types of VCS

- Centralized Version Control System (CVCS)

- Distributed Version Control System (DVCS)

Centralized VCS – CVCS

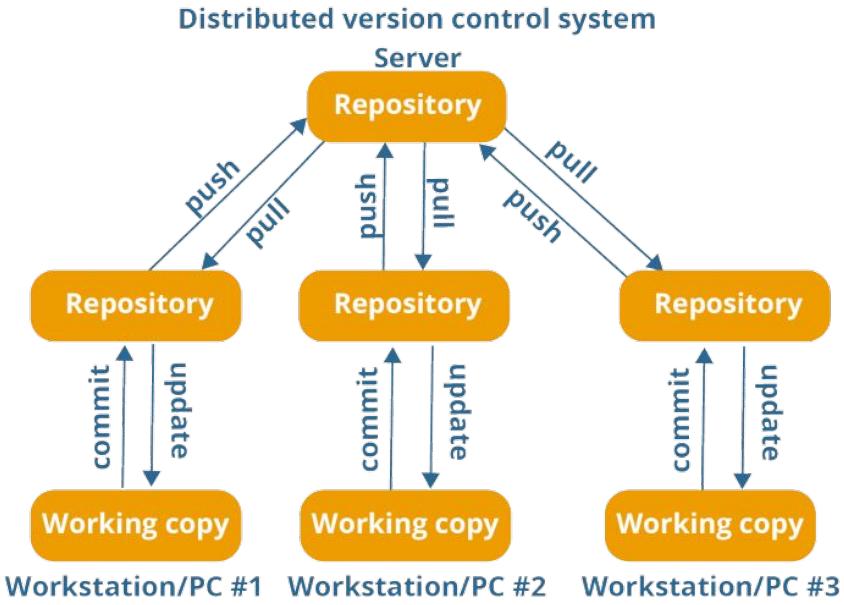
- uses a central server to store all files and enables team collaboration.
- works on a single repository to which users can directly access a central server.



- Here Each Work Station is connect with Central Code Repository.
- **Drawbacks** - It is not locally available. ☹
- Crash of CVCS will result in losing the entire data of the project. ☹

Distributed Version Control System

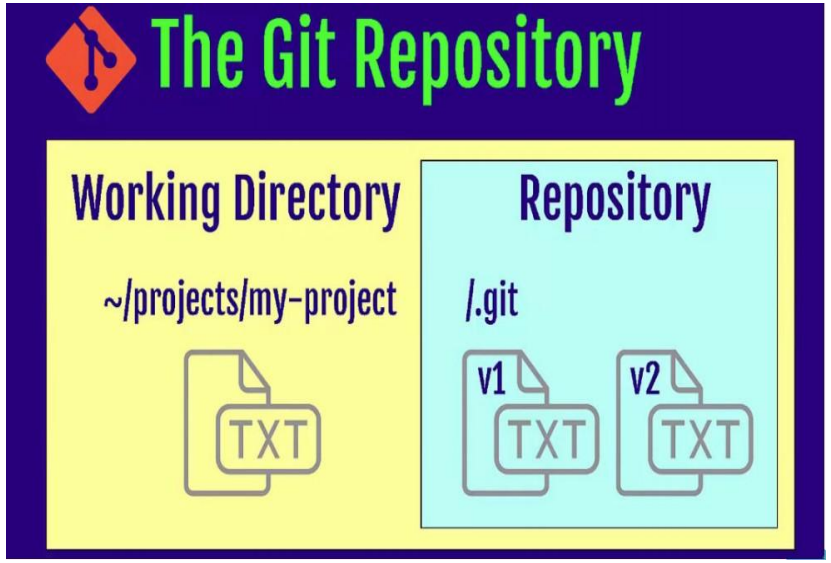
- In Distributed VCS, every contributor has a local copy or "**clone**" of the main repository.
- User can change and commit local Repo without any interference.
- User can update their local Repo from the Central Server.
- User can update the Central Server from their Repo.



- Operations in DVCS are fast.
- New changes can be done locally without manipulating the central data.
- If the central server gets crashed at any point of time, the lost data can be easily recovered from any one of the contributor's local repositories.

GIT Key Terminology

- Git Repository
  - GIT Repository contains Files, History, Config Managed by GIT.



GIT Key Terminology

- Working Directory - Area of Live Files, also known as Untracked area of GIT.
- Staging Area - Staging Area is draft space when git starts tracking and saving changes that occur in files.
- Git Directory - Also called 'local Repo', is your .git repo. It's area where GIT save everything.

GIT Key Terminology

- Remote repository is stored on a code hosting service like GitHub or on an internal server. In order to be synchronised between multiple users and to keep the work safe and secure.

- Branch in GIT - Branch in Git is a way to keep developing and coding a new feature or modification on the software and still not affecting the main part of the project.

LAB

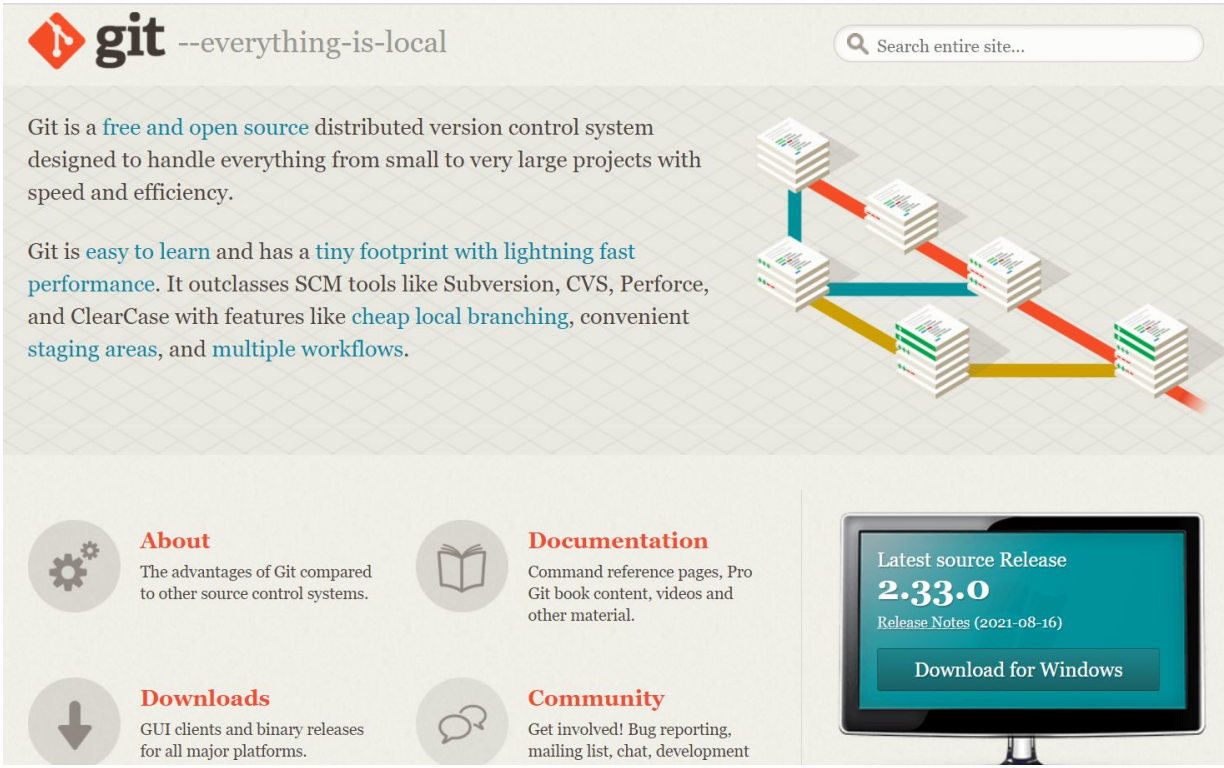
- Virtualization tool: Vmware Workstation Pro
- Objective: preparation of two virtual machines (CentOs) for next Labs
  - Virtual machine 1: Git, Docker
  - Virtual machine 2: Jenkins
- You have to create a virtual network for the interconnection of the two machines:
  - Private network 192.168.10.0/24
  - Automatic NAT network for internet access
- Use Mobaxterm to access the 2 VMs from the host machine

GIT Install

GIT is open source and can be installed on all major os

- Windows
- Mac OS
- Linux

Link : https://git-scm.com/



Download for Linux and Unix

It is easiest to install Git on Linux using the preferred package manager of your Linux distribution. If you prefer to build from source, you can find tarballs on kernel.org. The latest version is 2.33.0.

Debian/Ubuntu

For the latest stable version for your release of Debian/Ubuntu

```
# apt-get install git
```

For Ubuntu, this PPA provides the latest stable upstream Git version

```
# add-apt-repository ppa:git-core/ppa # apt update; apt install git
```

Fedora

```
# yum install git
```
(up to Fedora 21)

```
# dnf install git
```
(Fedora 22 and later)

Gentoo

```
# emerge --ask --verbose dev-vcs/git
```

Arch Linux

```
# pacman -S git
```

openSUSE

```
# zypper install git
```

Mageia

```
# urpmi git
```

Nix/NixOS

```
# nix-env -i git
```

For CentOS:

#sudo yum update

#sudo yum upgrade

#sudo yum install –y git

#git --version

Create Git project

#sudo mkdir /home/First_Git_Directory

#git init

#ls -al /home/First_Git_Directory/

Create Git project

Git stores the metadata and object database for the project in this .git directory like:

1. Remote information (to which remote server your project is connected)
2. History of all local commits
3. Branch information (on which branch is your current project state (HEAD) pointing to)
4. All logs of all local commits you have ever made (including revert changes)

- The .git folder is a bit like a magic hat into which you put your current magic show.
- Everything you organise into a show format is put inside this magic hat and can be 'pulled out' whenever, wherever you want.

Git: First Commit

#sudo touch file1.txt

#git status //Show the status of the working directory in the staging area

#ls -al /home/First_Git_Directory/

#git add file1.txt //to add file to the staging area

#git commit –m “my first file to commit” //keep your code updated with the latest changes and capture the state of the project at that point of time

Git: Tracking and Logs

#echo "hello second file" >> file2.txt //create and edit file2

#git status

#echo "add new line " >> file1.txt // modify in file1

#git status

#git add . //add all files in the current directory

#git commit –m “my second commit

Git: Tracking and Logs

#git log // show by default the last 10 commits

#git log –n // show the last n commits

git log --oneline //show log commit prettier

git log --author="email or name" // show the specific commit of the developer

GIT Configure User Information

#git help name_of_command

#git config --global user.name "name fmaily_name“

#git config --global user.email "name@xxx.com"

#git config --list

#git config --global color.ui auto

GIT Configure User Information

Why ?

- Just imagine a team of 100 developers modify on the same code and working each of them in the same time.

- **Account configuration is needed for more Team visibility.**

- To identify who has modified the source code and who is responsible of each commit
- the commit data determines which account name to attach to the push

GitHub

Github is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code.

GitHub is a Remote repository for your code base

github.com/new

Owner * **Repository name** *

tekup2021 / first_github_project

Great repository names are short and memorable. Need inspiration? How about **miniature-funicular**?

Description (optional)

this project for training purpose

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

- [x] **Add a README file**
This is where you can write a long description for your project. Learn more.

- [x] **Add .gitignore**
Choose which files not to track from a list of templates. Learn more.

.gitignore template: **GitBook**

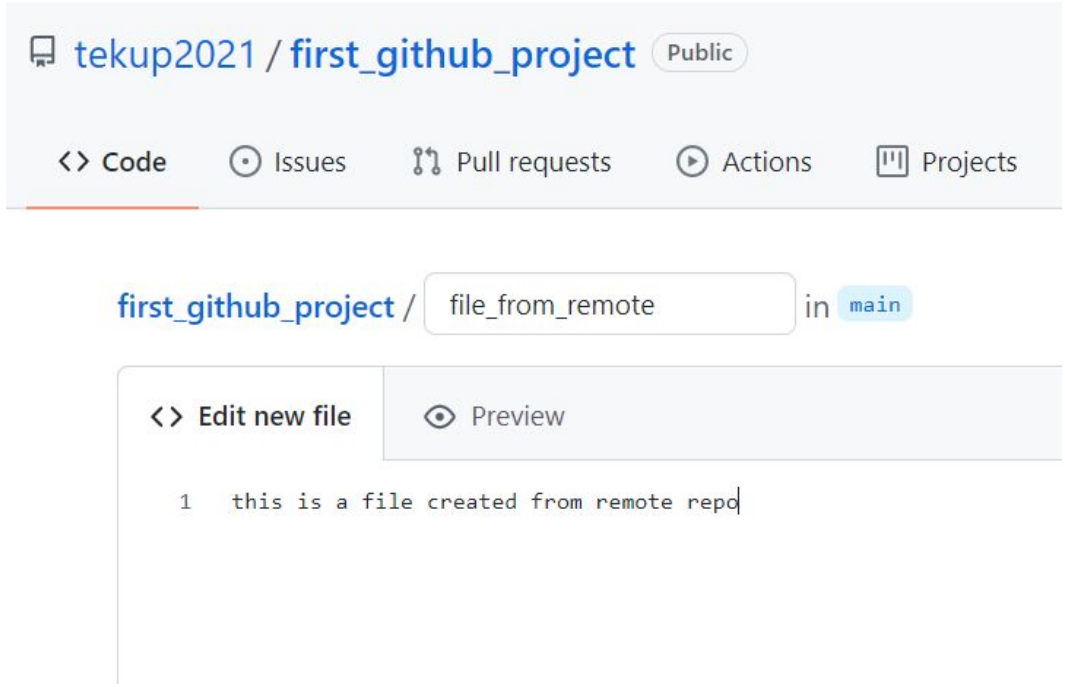
- [ ] **Choose a license**
A license tells others what they can and can't do with your code. Learn more.

This will set `main` as the default branch. Change the default name in your settings.

Create repository

GitHub

#git clone httpd://github.xxx/xxx.git



#git pull

GitHub

#echo "create file from local repo to send" >> local_file.txt
#git add .
#git commit –m “local file to send “
#git push
>> remote: Support for password authentication was removed on August 13, 2021. Please use a personal access token instead.

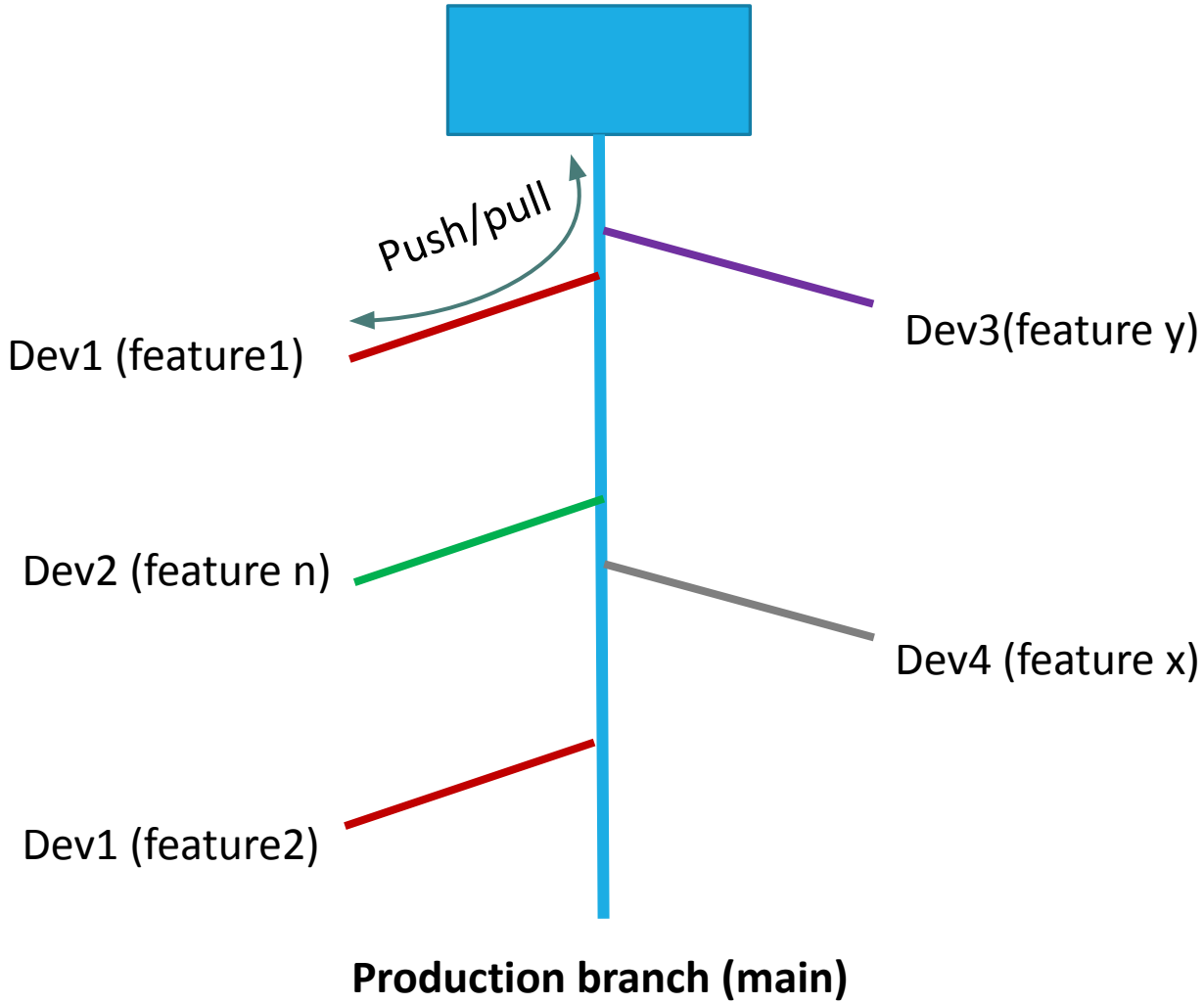
To generate our personal token from github:
Settings > Developer settings > Personal access tokens
Generate a token to use it instead of the password

SETUP Authentication in local with GitHub

SSH Connection steps:

1. Open the github settings> ssh and gpg keys
2. Open your terminal and execute the command
   - # ssh-keygen -t ed25519
   - Hit enter to create your ssh key
   - Copy public key and paste it in github ssh key creation
3. #git clone git@github.com:tekup2021/ssh_repo.git
4. #echo 'hello ssh repo' >> file1.txt
5. #git add .
6. #git commit -m "my first file in ssh repo connection with github"
7. #git push

Git: Concept of branches

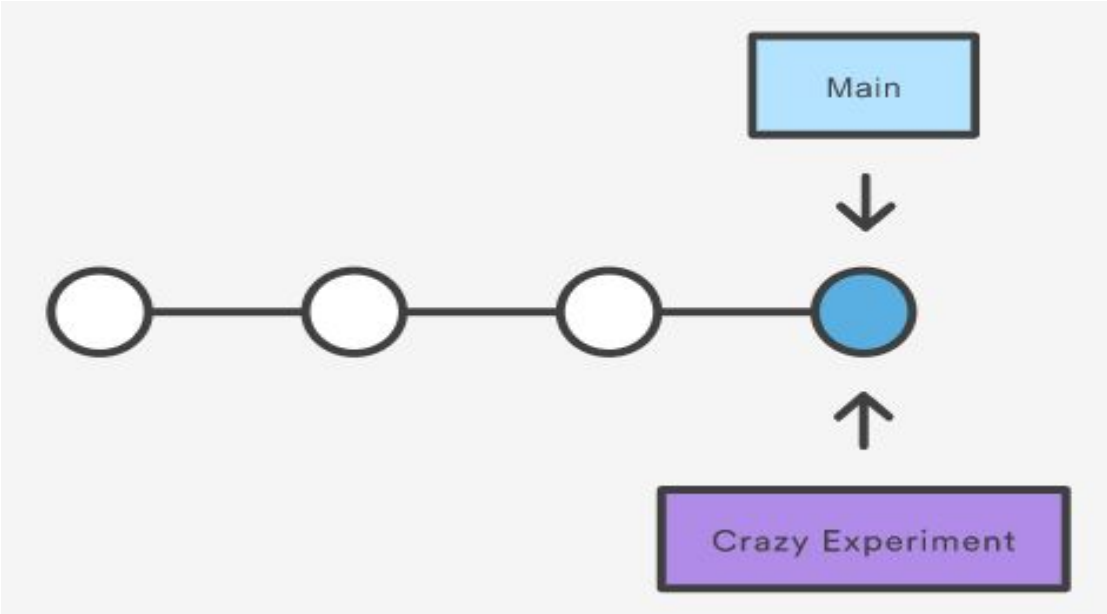


Git: Concept of branches

- A branch represents an independent line of development.
- It's important to understand that branches are just pointers to commits. When you create a branch, all Git needs to do is create a new pointer, it doesn't change the repository in any other way
- you start with a repository that looks like this:

Git: Concept of branches

- Then, you create a branch:
  The repository history remains unchanged. All you get is a new pointer to the current commit



Git: Concept of branches

- To list all branches
  - #git branch –a
- To create new branch
  - #git branch branch_name
- To rename branch
  - #git branch –m old_name new_name
- To delete branch
  - #git branch –d branch_name
- To switch to specific branch
  - #git checkout branch_name

Git: Concept of branches

Manipulating branch steps:

1. #git branch
2. #git branch feature1
3. #git branch
4. #git checkout feature1
5. #git branch

- To create branch and switch directly into it you can use

#git checkout –b feature2

6. Try to create different files into these branches and compare it with main branch

- For a new branch we have to push using #git push origin feature
- Before pulling this branch from remote #git branch --set-upstream-to=origin/feature

Git: Compare

To compare **working directory** with **staging area** after modification:

- # echo 'diff newline' >> file_name
- # git status
- #git diff file_name //to see what modification is added or deleted

To compare **commits in git**

- #git log –oneline
- #git diff **older_commit newer_commit** //to detect exactly what is recently added to older_commit
- #git diff commitA commitB **–name-only** //to list just the name of files impacted

Git: Compare

To compare **working directory** with **staging area** after modification:

- # echo 'diff newline' >> file_name
- # git status
- #git diff file_name //to see what modification is added or deleted

To compare **commits in git**

- #git log –oneline
- #git diff **older_commit newer_commit** //to detect exactly what is recently added to older_commit
- #git diff commitA commitB **–name-only** //to list just the name of files impacted

Git: Undoing Changes

Roll Back is all about to Undo the changes, you did in repo.

- In GIT this can be done via RESET and REVERT.
- RESET - Practically, user can think of it as a “rollback”.
- Reset command points local environment back to a previous commit.

Git: Undoing Changes

REVERT - Net effect of the ***git revert*** command is similar to reset, but its approach is different.

Revert adds a new commit at the end of the chain to "**cancel**" changes.

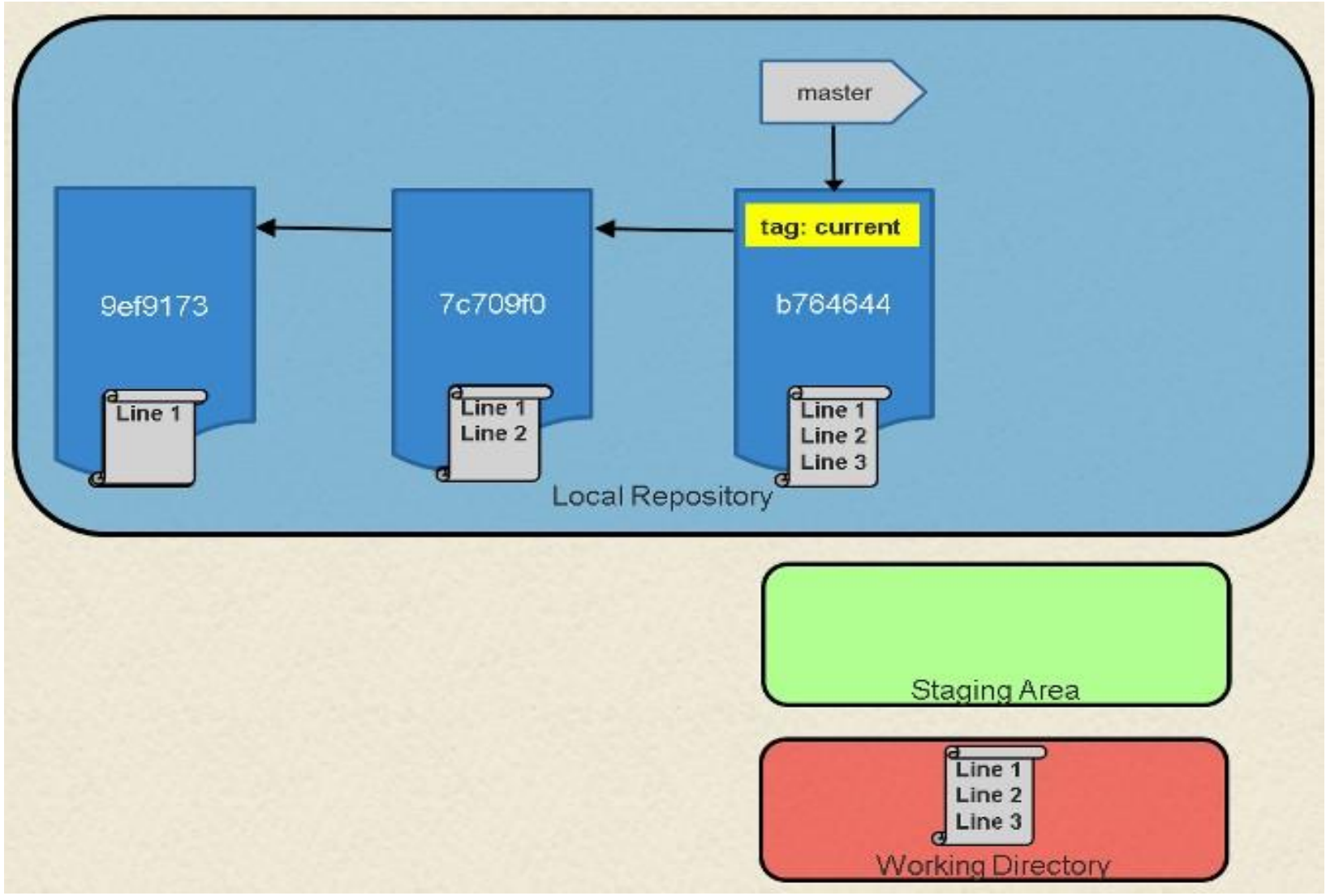
Revert or Reset ?

If user have already pushed commits to the remote repo, a **revert** is a nicer way to cancel out changes.

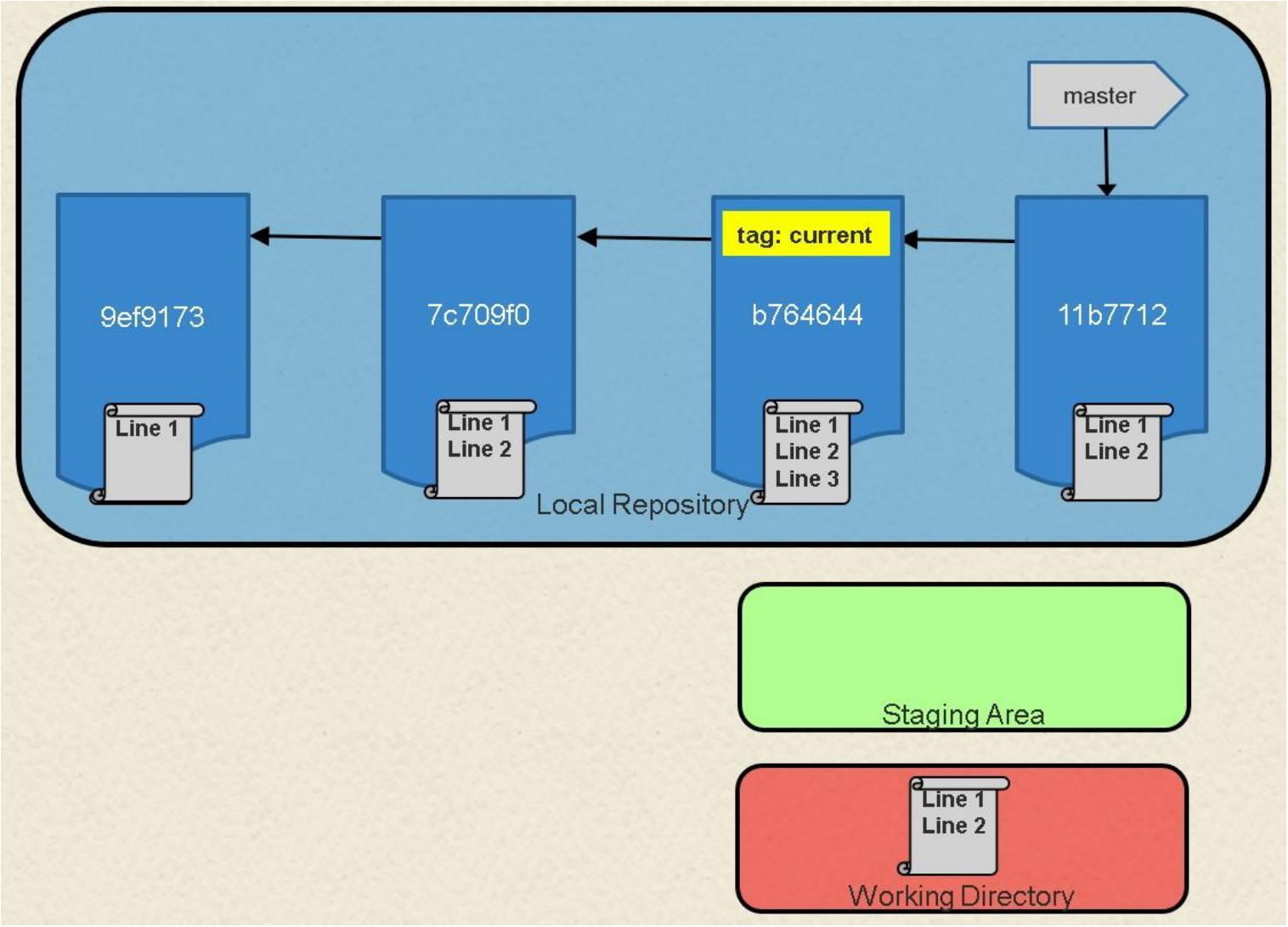
Git workflow works well for picking up additional **commits** at the end of a branch, but it can be challenging if a set of commits is no longer seen in the chain when someone resets the branch pointer back.

If Commit in local then Reset is good, If commit is pushed then revert is good option.

Git: Undoing Changes



Git: Undoing Changes



Git: Undoing Changes

To **Hard reset** your changes follow the steps:

- #echo 'wrong code' > file_to_reset.txt
- #git add .
- #git commit -m "commit to reset"
- #git push
- # git log --pretty=oneline --decorate
- #git reset –hard id_of_commit // or your can use #git reset --hard HEAD~N (N represent the number of commit your want to delete)
- #git log –oneline //to verify that your have successfully reseted your commit

Git: Undoing Changes

Note:

If you have done a mistake by reseting more commits than you were supposed to do

- #git reflog //to detect the id of the commit you have deleted (suppose it is XXX)
- #git reset –hard XXX
- #git log –oneline // to verify that you have recovered your commits

Git: Undoing Changes

To **Soft reset** your changes follow the steps:

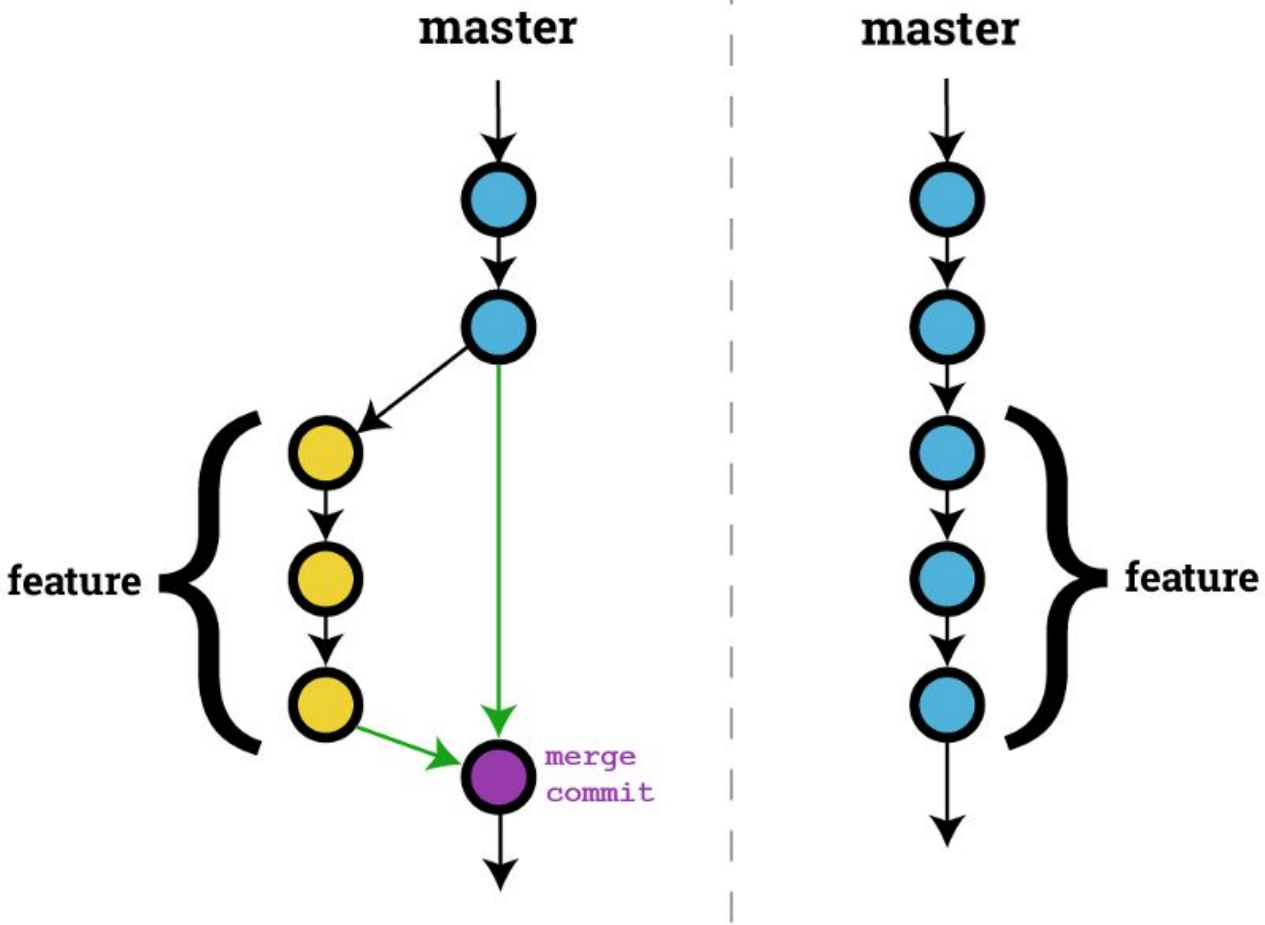
- # git reset –soft id_of_commit
- #git status //notice there are files ready to commit and waiting in the staging area
- # git reset file_name //to cancel the git add and let files become untracked by git
- #git status //to verify that your files are in the untracked area now.
- Correct your mistake by modifying your files and then add,commit and push them

Git: Undoing Changes

To **revert** your changes follow the steps:

- # git revert HEAD //it will ignore your last commit and placing a new commit upon it
- # git revert --no-commit HEAD // to check what files are going to be changed/deleted before confirming commit
- # git revert --no-commit HEAD~3.. //to ignore the last three commit at once
- # git revert --no-commit xxyyyzz..HEAD //to ignore all commit for the xxxyyyzzz commit until HEAD
- # git revert –abort // to cancel the revert
- #git revert –continue // to validate revert and writing commit message

Git merge



- Open github ui and go to the section **pull request** to apply your merge
- Or do it with git commande
  - #git checkout main
  - #git merge feature

Git merge

- Merging is a common practice for developers. Whether branches are created for testing, bug fixes, or other reasons, merging is commiting changes to another branch. It takes the contents of a source branch and integrates it with a target branch.

- Simply it integrates changes from one branch into another.

Git merge

To **merge branches** follow the steps:

- #git checkout –b hotfix
- #echo 'resolve production problem' >> hotfix.txt
- #git add .
- #git commit –m "this is correction in the hotfix branch"
- Switch to the other main branch and do 2 commits
- In the same main branch merge the hotfix modification into main source code :
- #git merge hotfix
- #git log –oneline –graph --decorate

Git merge

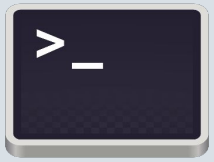
The merge keeps the commits history as it is. So the branch in which we had executed the merge will be badly organized

- #git merge hotfix --squash
- #git status
- #git commit –m “commit encapsulating all hotfix commits branch”

Git: merge conflict

Let’s invoke a **merge conflict** to resolve it:

- #git checkout -b feature_db
- #echo "this is featuredb source code" > data_base.db
- #git add .
- #git commit -m "feature modif for db connection"
- #git checkout main
- #echo "this is main source code" > data_base.db
- #git status
- #git add .
- #git commit -m "main modif for db connection"
- #git merge feature_db




```
Auto-merging data_base.db
CONFLICT (add/add): Merge conflict in data_base.db
Automatic merge failed; fix conflicts and then commit the result.
```

Git: Resolving merge conflict

When you Open the data_base.db file you will find:

```
<<<<<<< HEAD
this is main source code
=======
this is featuredb  source code
>>>>>>> feature_db
```

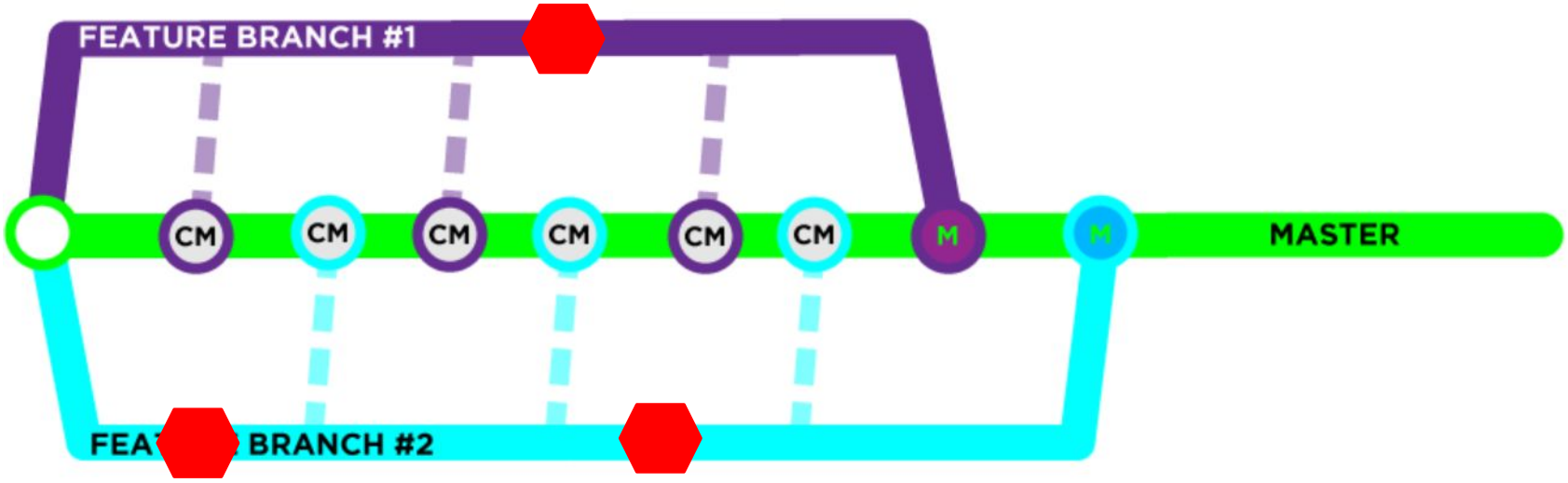
Keep the piece of code that you want to have in you main branch by erasing also

```
=======
<<<<<<< HEAD
>>>>>>> feature_db
```

- save your file
- #git add
- #git commit -m "resolving merge conflict“
- #git log --oneline --graph --decorate

Git: Resolving merge conflict

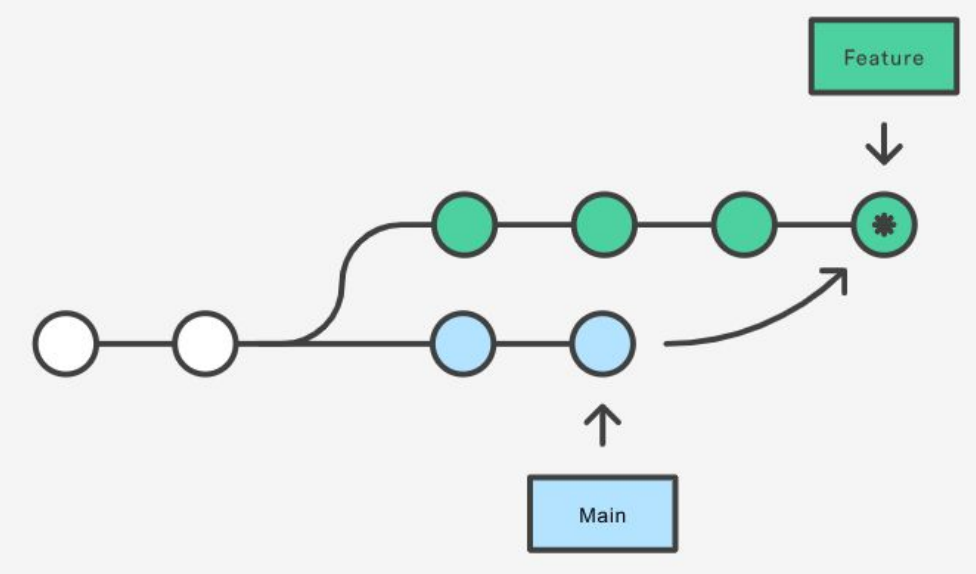
Merge Workflow to avoid resolving conflict in master branch

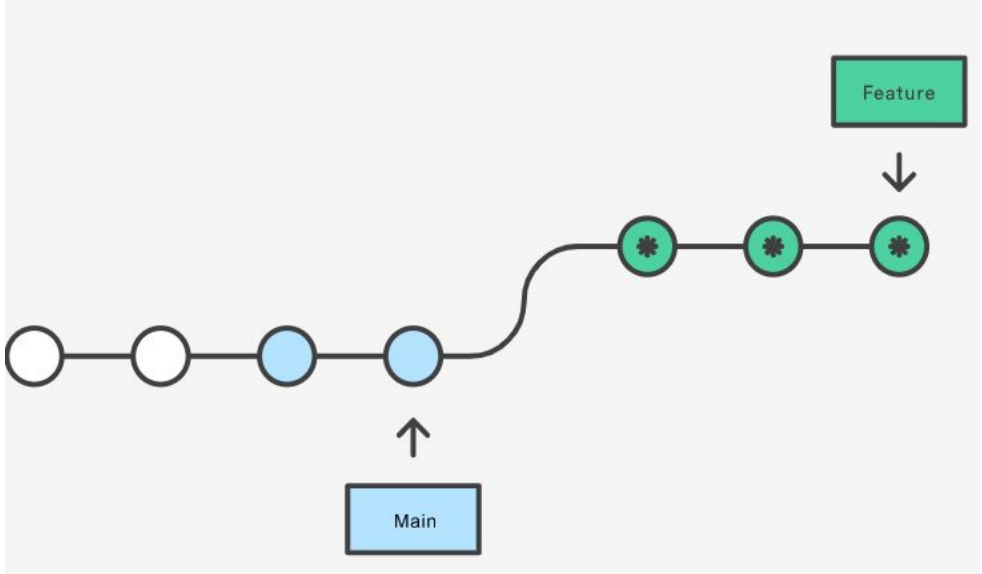


You have to continiously merge the master branch to resolve conflicts if it's happen in the feature branch not the main branch

Git: Rebase & Merge

- **Git rebase vs merge**, both actions can be used to combine file changes from one branch to another.
- What is Git rebase? Rebase is an action in Git that allows you to rewrite commits from one branch into another branch
- instead of creating a merge commit, rebase consists of rewriting the history of the project by creating new commits for each commit of the original branch





Git: Rebase

#git rebase branchA branch

- Git will check and detect the last commit have both branches in **commun**
- Git has a look at the **current branch** (branchA) and see what **changed** actually in this branch
- Git **saves** these changes (commits) **internally**
- Git go back to the **other branch** (branchB) and see what **changes** are applied after the last common commit
- Git **take these changes** of the other branch (branch) and go back to the current branch (branchA) to **apply** it **on top of the commun commit**
- Finally Git apply the changes that **have been saved internally** on the **top**

Git: Stashing

#git stash // just saves the state of staging area

#git stash –a // saves state of staging area and untracked area

#git stash list //show stashes

#git stash apply //applies last stash and saved in stash

#git stash pop //aplie but erase it from saved stashes

Git:

Cherry-pick

Cherry-pick is the copy of a useful specific commit from branch to another

#git Cherry-pick id_of_commit_of_the_branch_we_want_to_copy_from

Module Intégration Continue

History

- In 2004, appearance of Hudson, an open source continuous integration solution developed by Sun MicroSystems
- After a conflict (the use of the name Hudson by Oracle), in November 2010, proposal to rename the project to Jenkins
- In February 2011, Oracle decided to continue to develop Hudson. Jenkins is considered a fork of Hudson
- Today Jenkins is the only survivor!

What is Jenkins

- Jenkins is an open-source Continuous Integration server written in Java for orchestrating a chain of actions to achieve the Continuous Integration process in an automated fashion.

- Jenkins supports the complete development life cycle of software from building, testing, deploying the software.

- Jenkins is a feature Rich Application, which help organization in fast s/w delivery.

What is Jenkins

CI / CD solution written in Java and allowing:

- Automate builds and tests by configuration or scripting.
- Monitor the construction phases of projects
- To build and deploy on remote instances (master / salve)
- To be able to easily obtain the binaries of the latest stable versions

Solution not specific to Java projects. Ability to use Jenkins with many languages (javascript, php, .net, c / c ++, swift ...)

What is Jenkins

- Code Built and Test as soon as Developer Commit the Changes.
- Auto Deployment on **Successful** Software Build and Notify Stakeholders.
- Notify Errors to Development team on Build Failure.
- Automating Build Process saving the Delivery Time and reduce the Defects.

What is Jenkins

- Des centaines de plugins disponibles.
- Facilement extensible
- Possibilité de définir finement des droits d’accès par projet

Why Jenkins

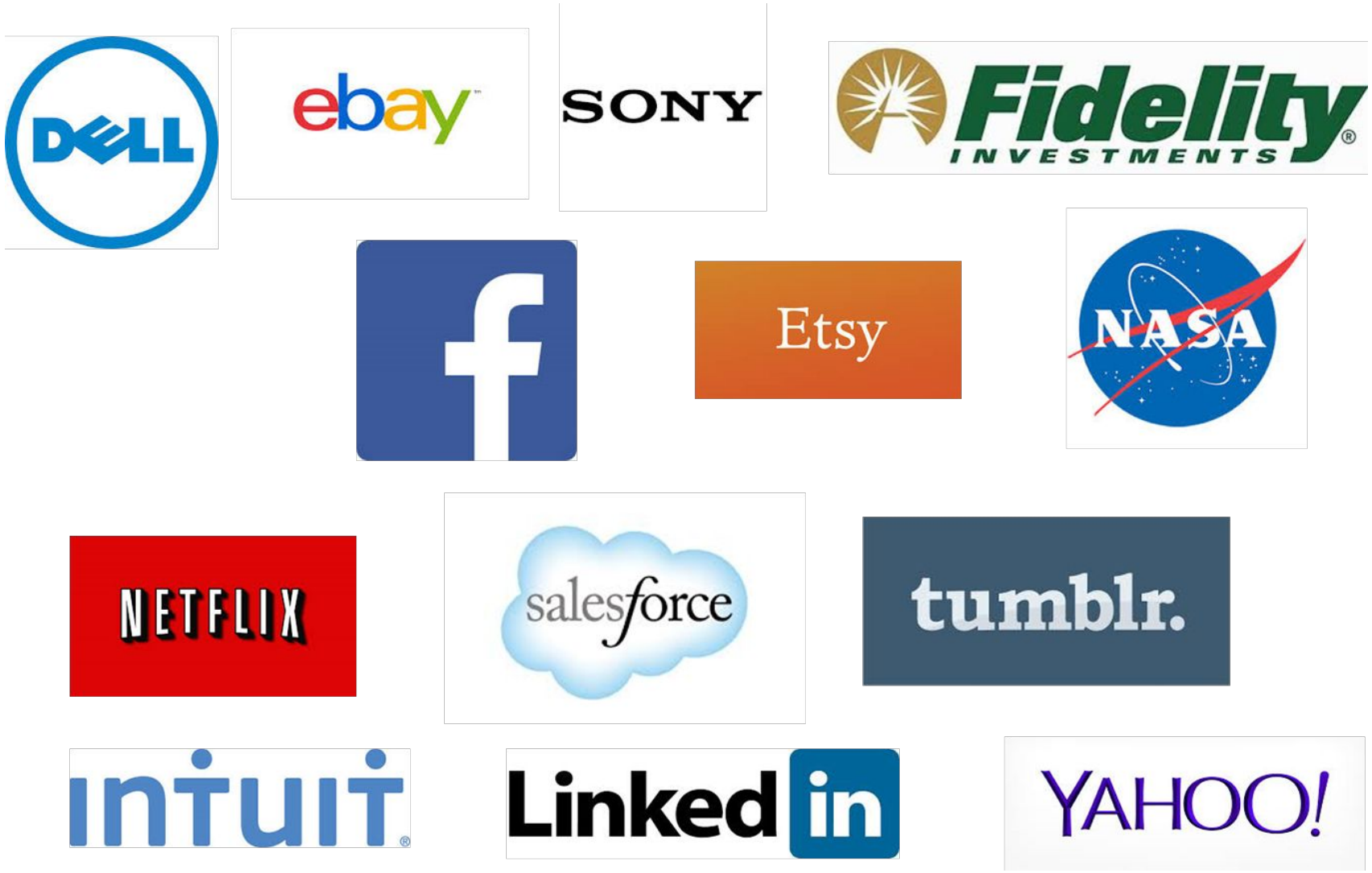
Let us imagine, that there are around 10 developers who are working on a shared repository. Some developer completes their task in 25 days while others take 30 days to complete.

Before Jenkins	After Jenkins
Once all Developers had completed their assigned coding tasks, they used to commit their code all at same time. Later, Build is tested and deployed. Code commit built, and test cycle was very infrequent, and a single build was done after many days.	The code is built and test as soon as Developer commits code. Jenkin will build and test code many times during the day If the build is successful, then Jenkins will deploy the source into the test server and notifies the deployment team. If the build fails, then Jenkins will notify the errors to the developer team.

Why Jenkins

Before Jenkins	After Jenkins
It is not an easy task to isolate, detect, and fix errors for multiple commits.	Since the code is built after each commit of a single developer, it's easy to detect whose code caused the built to fail
Code build and test process are entirely manual, so there are a lot of chances for failure.	Automated build and test process saving timing and reducing defects.
The code is deployed once all the errors are fixed and tested.	The code is deployed after every successful build and test.
Development Cycle is slow	The development cycle is fast. New features are more readily available to users. Increases profits.

Who Use Jenkins



Jenkins Key Terminology

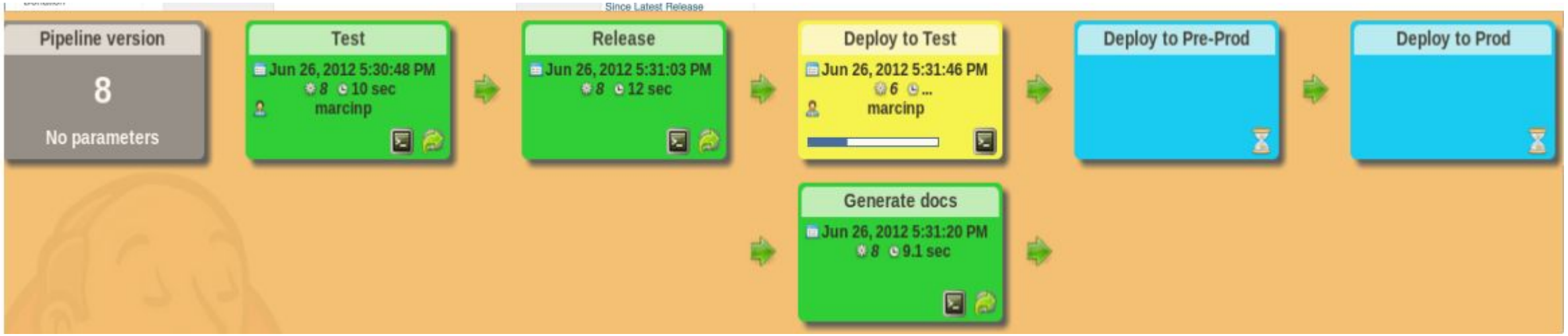
- **Job:** is the definition of an executable task or a structure to organize our Jenkins projects.
- **View & Folder:** Are options in Jenkins used to regroup jobs by category depending on what structure you want to work on.
- **Extension & Plugin:** are the primary means of enhancing the functionality of a Jenkins environment to suit the user-specific needs. can be installed on jenkins to provides multiple new fonctionnalities on (build tool, test tool, cloud providers, analyse tool..etc)

Jenkins Key Terminology

- **Master**: is the Jenkins main server coordinating processes such as configuration storage, plugin management, and UI display.
- **Agent**: is a slave machine orchestrated by the master in order to balance the workload between nodes and executes tasks en parallel
- **Dashboard:** is a simple and powerful place where we can manage all jobs and therefore manage the application delivery pipeline as well

Jenkins Key Terminology

- **Pipeline:** is an automated expression of your process for getting software from version control right through to your users and customers. enable you to define the whole jobs workflow based on script written with DSL (domain-specific language)



Jenkins Key Terminology

Push to Harbor - 2m 11s

重启 Push to Harbor

docker login -u $username -p $password harbor.guojiaxing.red — Shell Script <1s

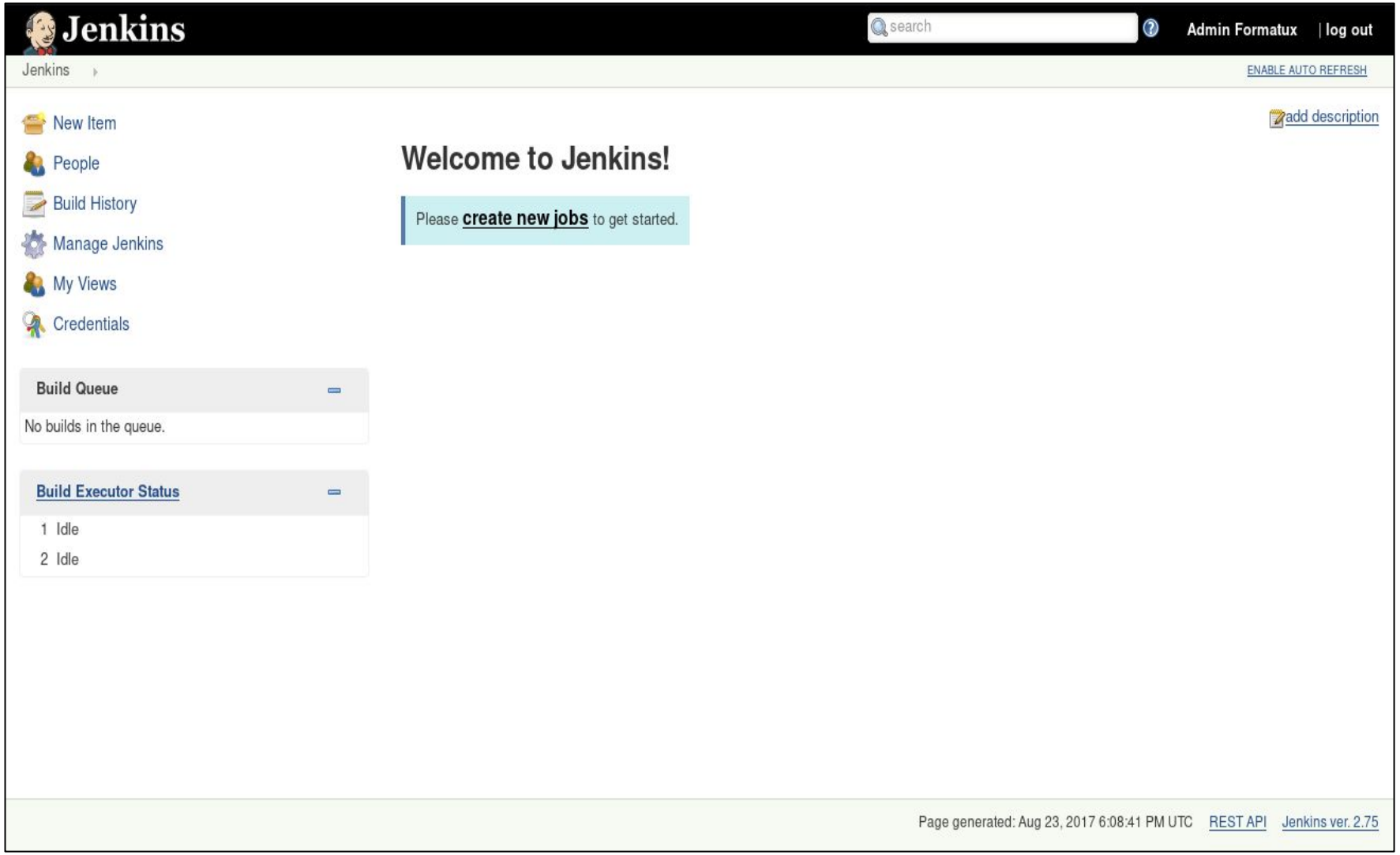
```
+ docker login -u **** -p **** harbor.guojiaxing.red
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

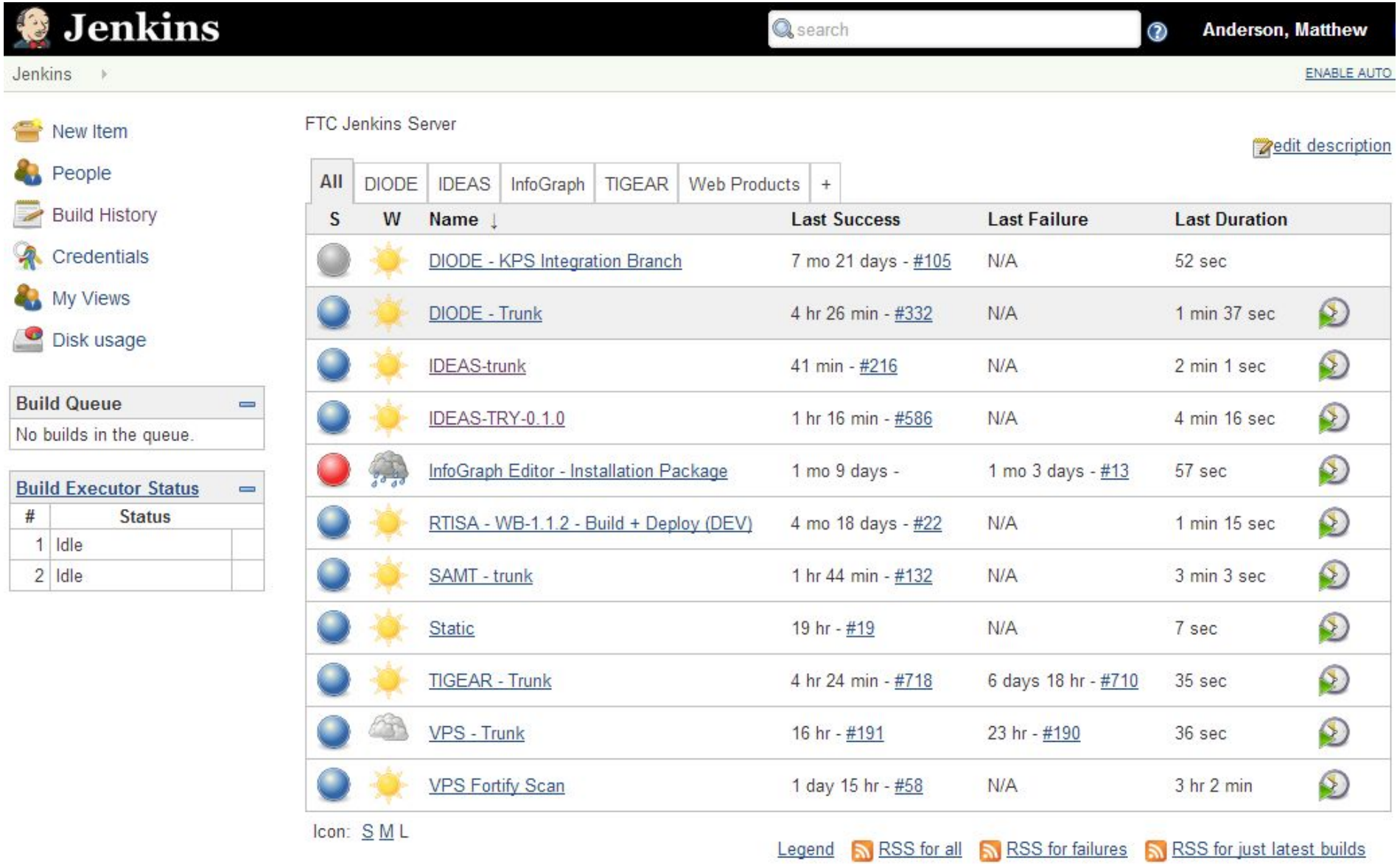
export VERSION=$(cat ${WORKSPACE}/VERSION) echo "docker push ${IMAGE_NAME}_${BRANCH_NAME}:${VERSION}" docker push ${IMAGE_NAME}_${BRANCH_NAME}:${VERSION} — Shell Script 2m 10s

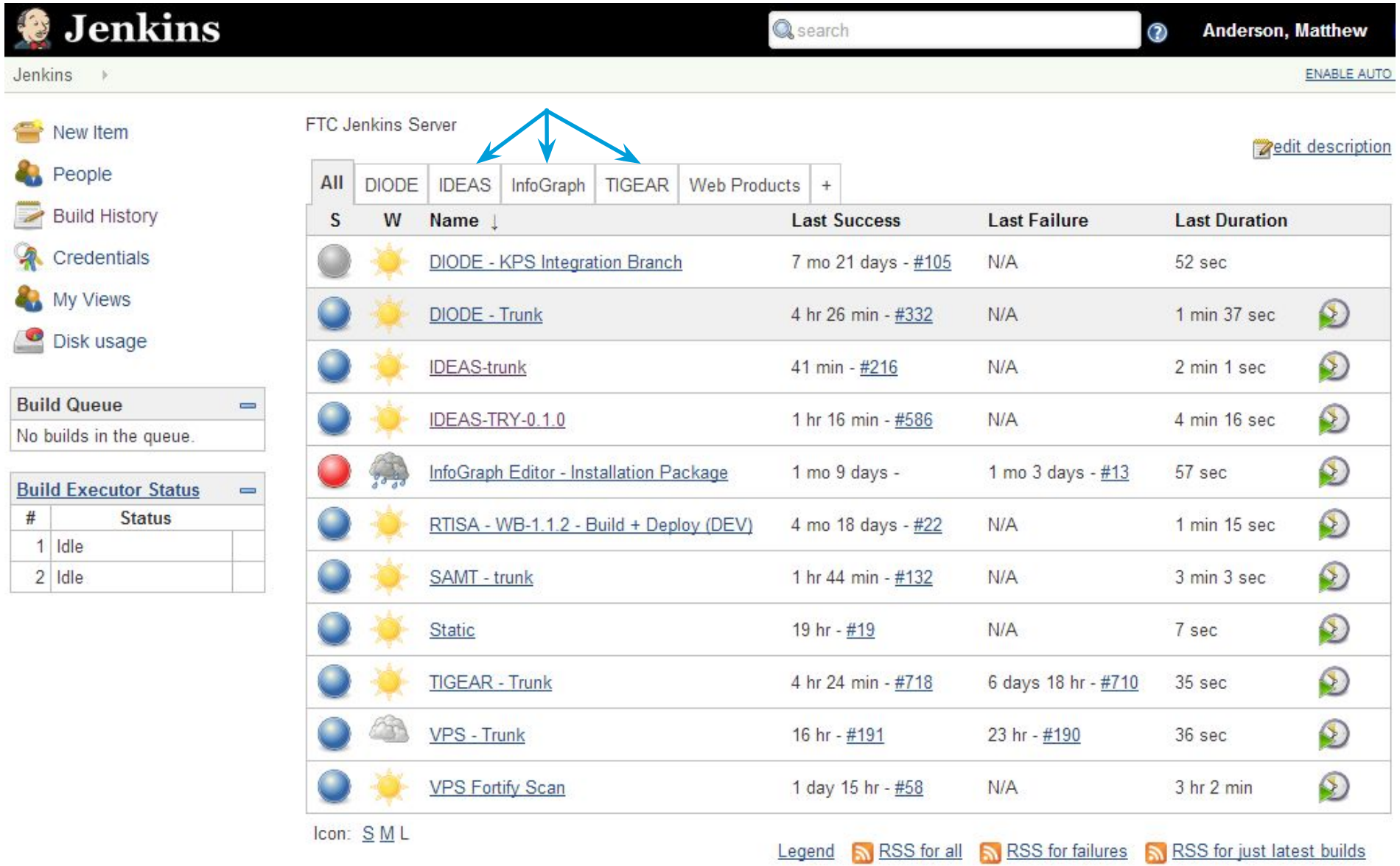
environmental_clean(){docker_ps=`docker ps | grep ${CONTAINER_NAME}_${BRANCH_NAME}`docker_psa=`docker ps -a | grep ${CONTAINER_NAME}_${BRANCH_NAME}`if [[0 -eq $docker_ps]];then... — Shell Script 1s

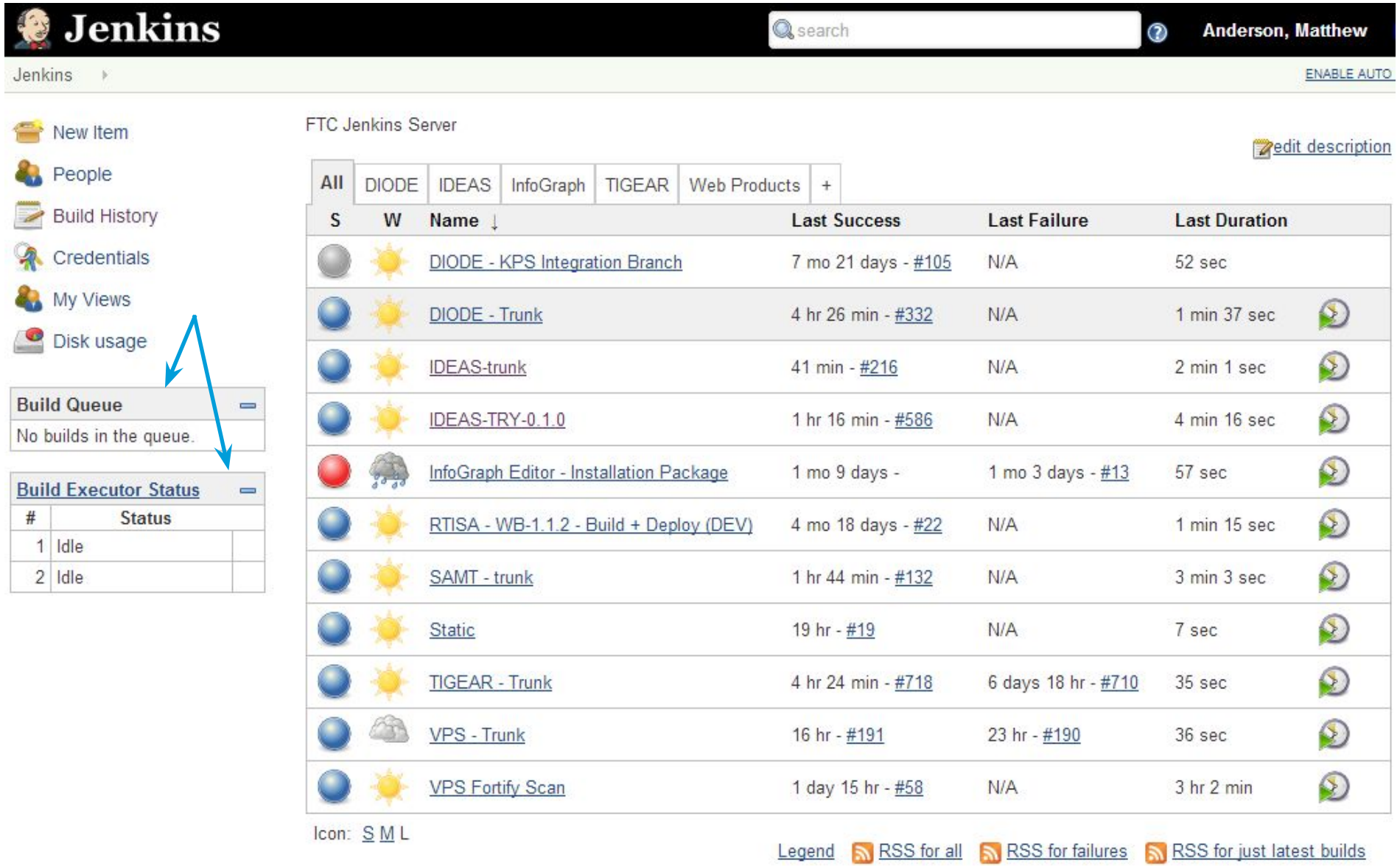
Extended Email 3s

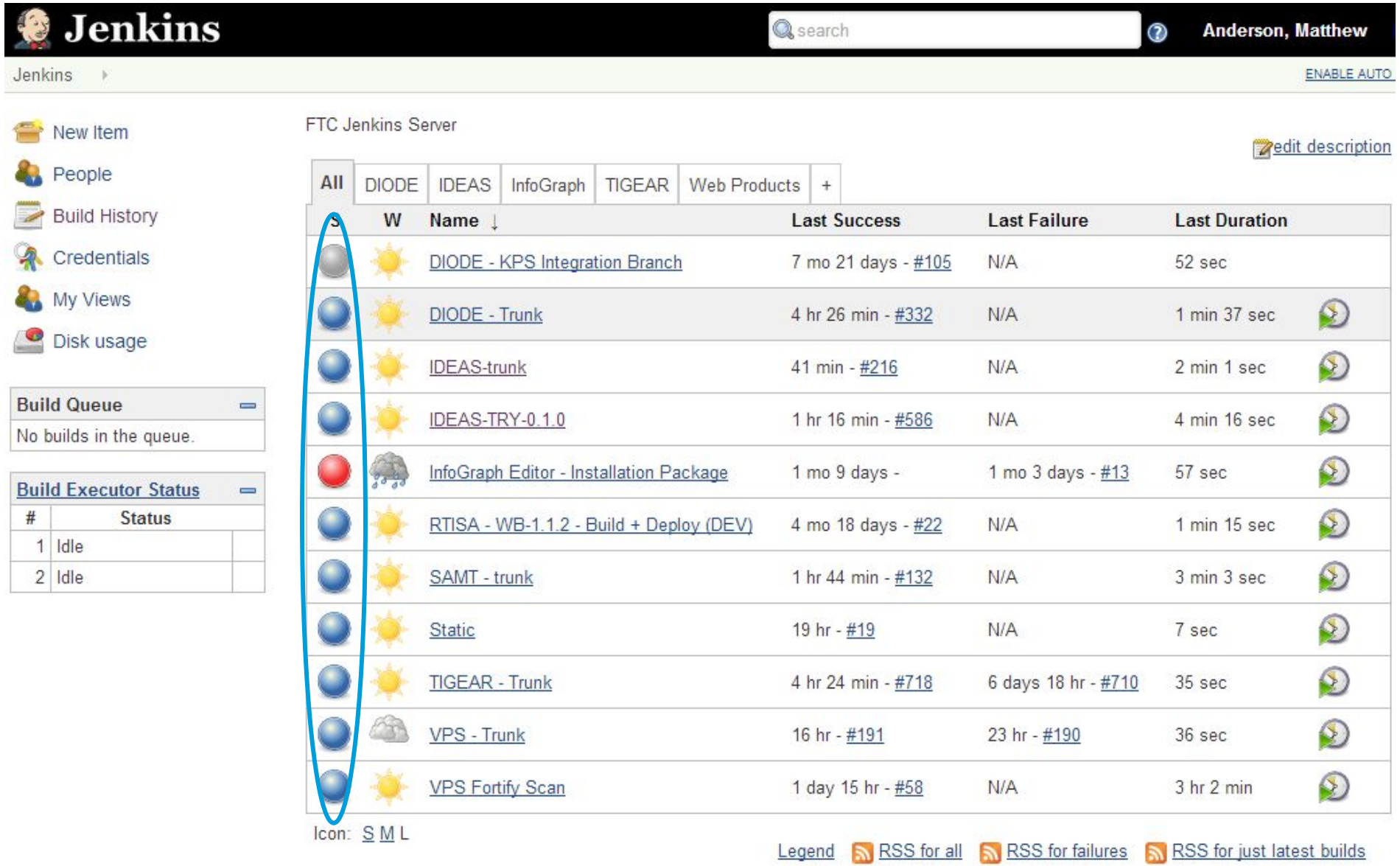
Jenkins Dashboard

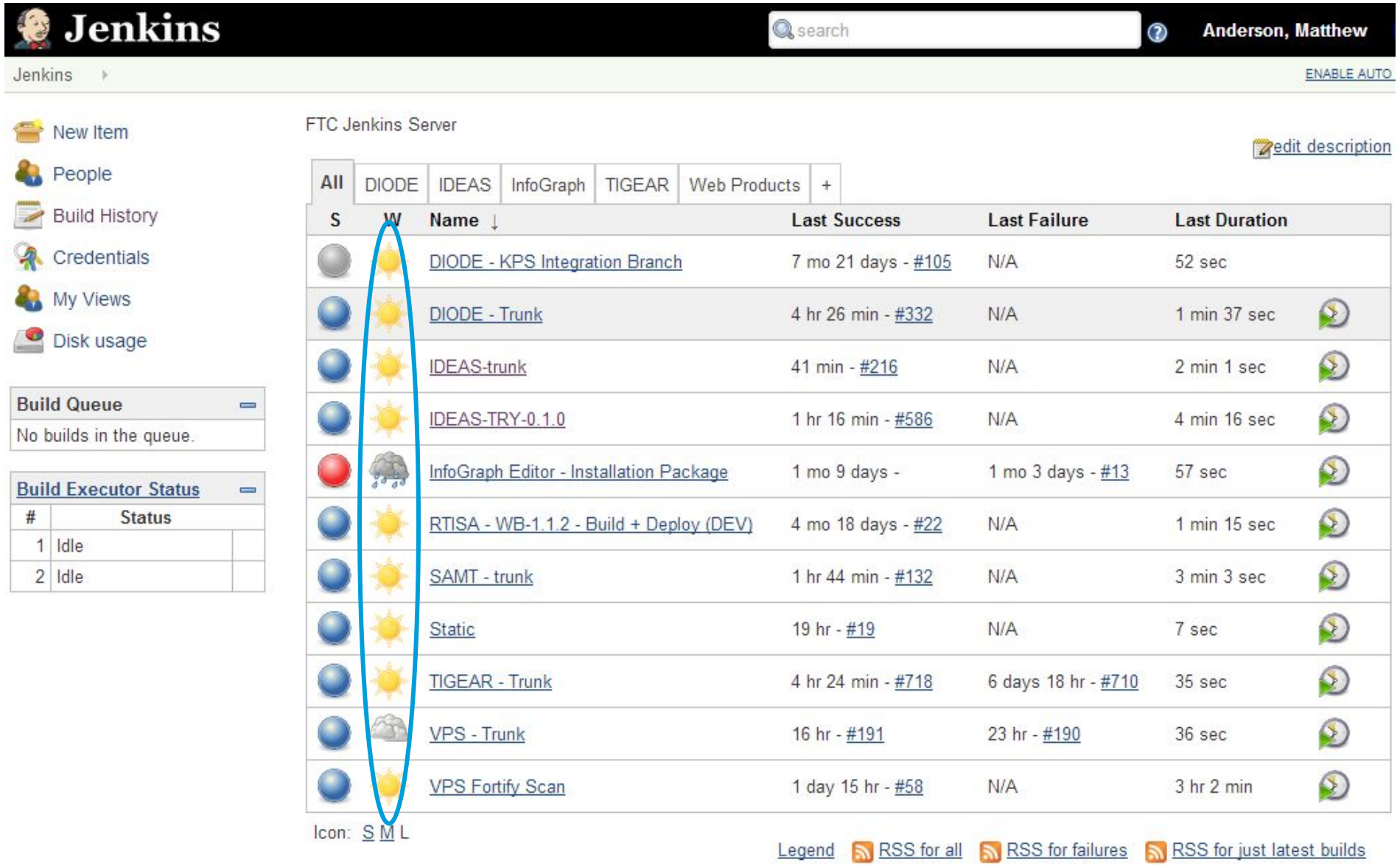


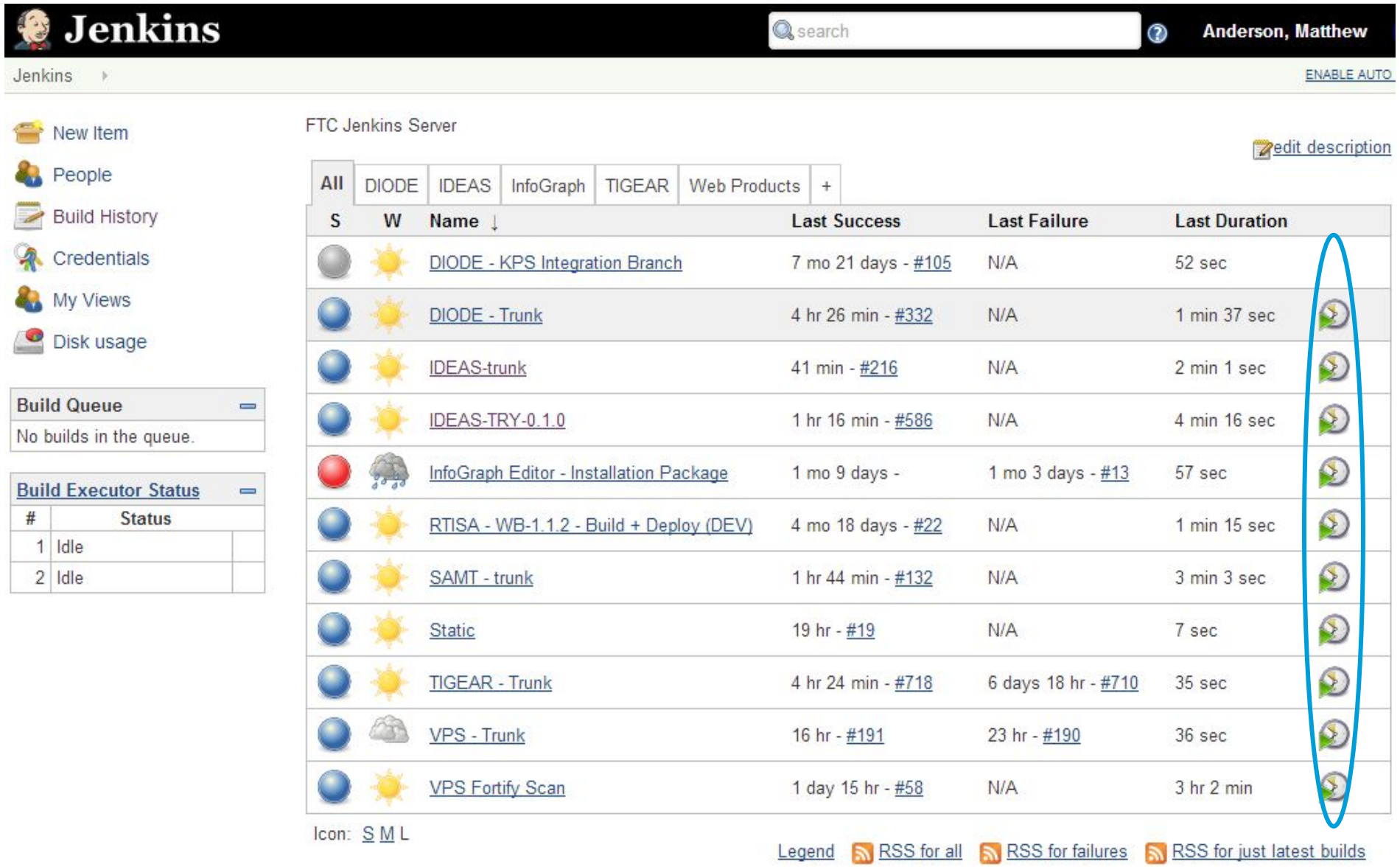

Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All DIODE IDEAS InfoGraph TIGEAR Web Products +
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend RSS for all RSS for failures RSS for just latest builds


Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All
DIODE
IDEAS
InfoGraph
TIGEAR
Web Products
+
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend
RSS for all
RSS for failures
RSS for just latest builds

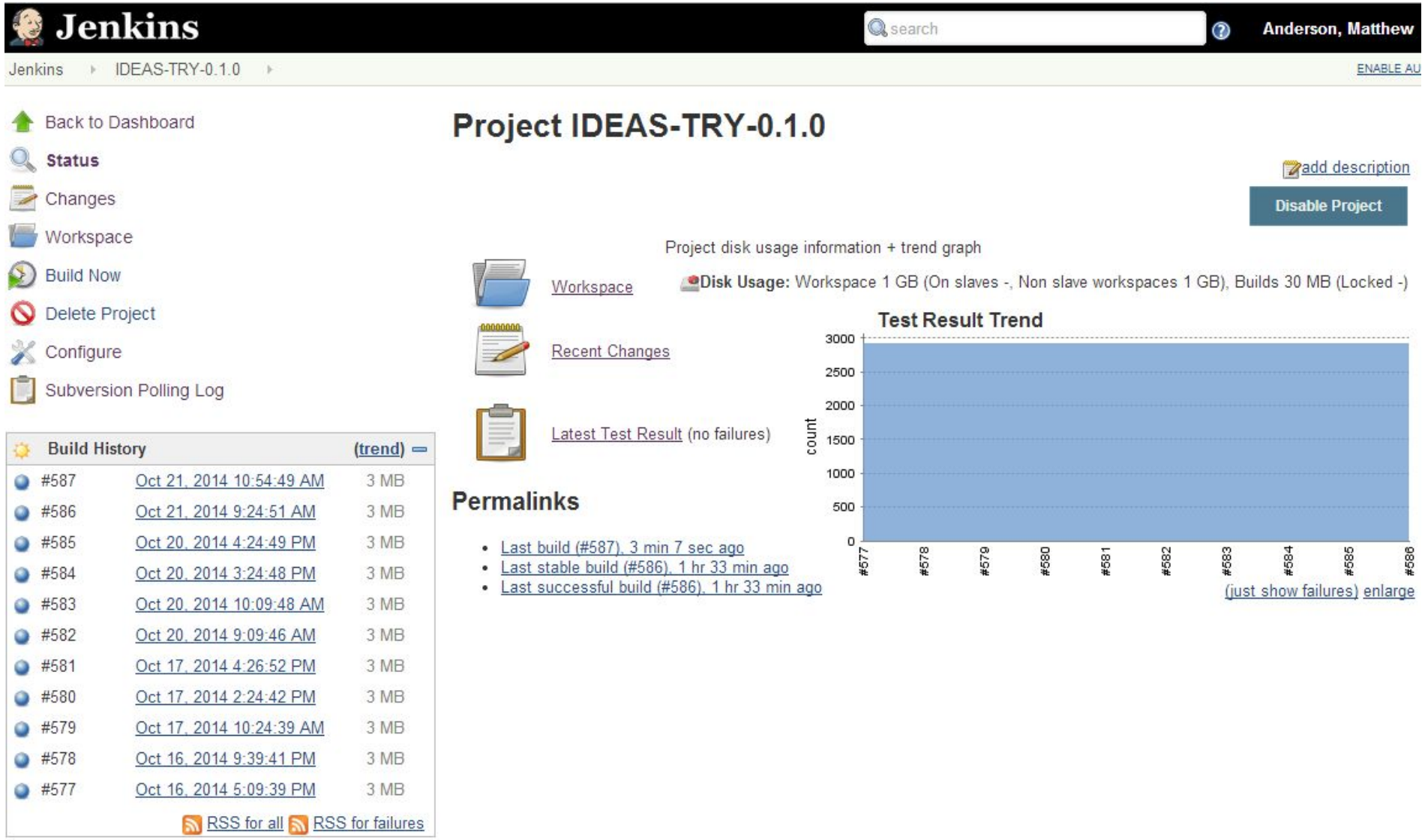

Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All DIODE IDEAS InfoGraph TIGEAR Web Products +
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend RSS for all RSS for failures RSS for just latest builds


Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All DIODE IDEAS InfoGraph TIGEAR Web Products +
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend RSS for all RSS for failures RSS for just latest builds

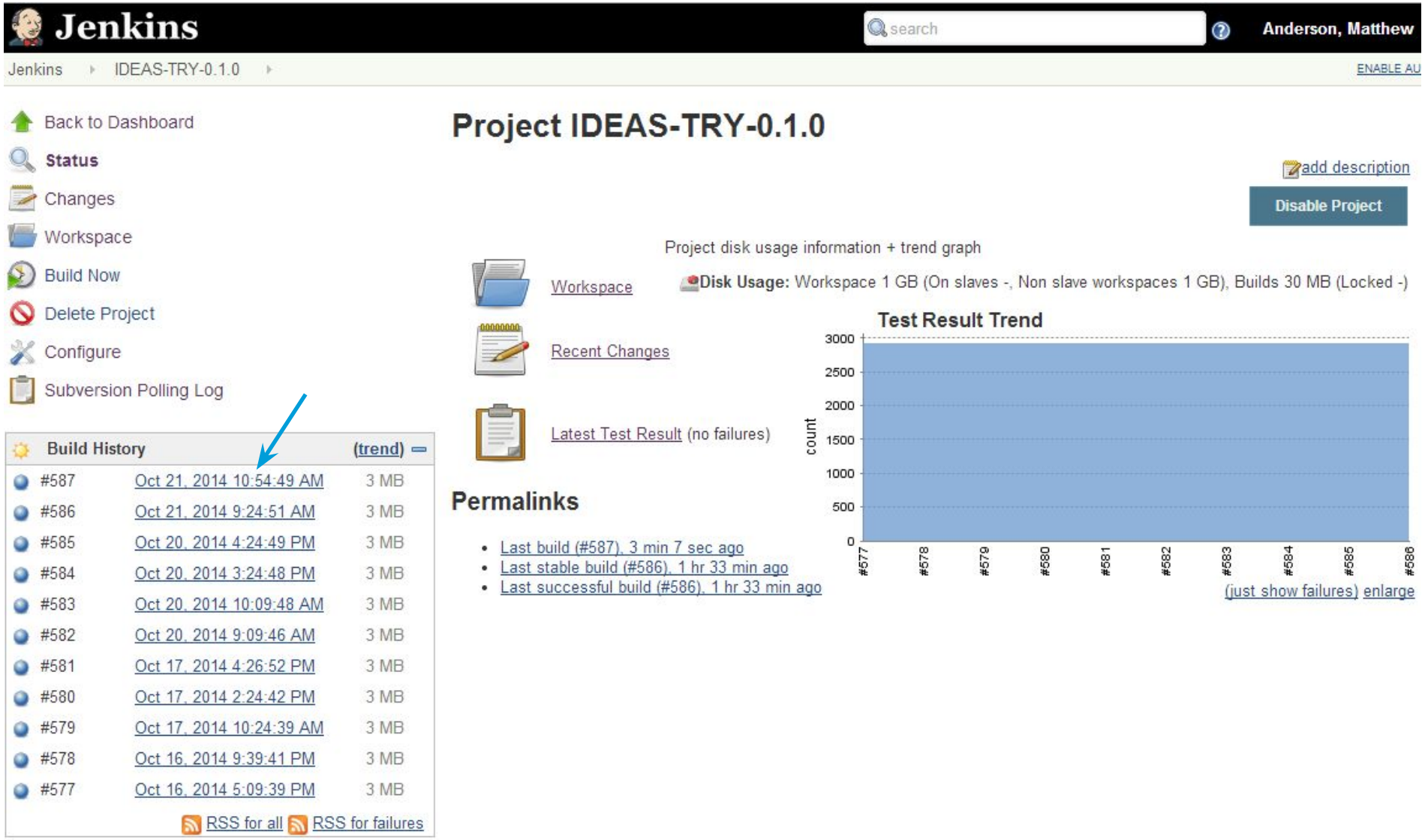

Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All DIODE IDEAS InfoGraph TIGEAR Web Products +
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend RSS for all RSS for failures RSS for just latest builds


Jenkins
search
Anderson, Matthew
Jenkins
ENABLE AUTO
New Item
People
Build History
Credentials
My Views
Disk usage
Build Queue
No builds in the queue.
Build Executor Status
Status
1 Idle
2 Idle
FTC Jenkins Server
edit description
All DIODE IDEAS InfoGraph TIGEAR Web Products +
S W Name ↓ Last Success Last Failure Last Duration
DIODE - KPS Integration Branch 7 mo 21 days - #105 N/A 52 sec
DIODE - Trunk 4 hr 26 min - #332 N/A 1 min 37 sec
IDEAS-trunk 41 min - #216 N/A 2 min 1 sec
IDEAS-TRY-0.1.0 1 hr 16 min - #586 N/A 4 min 16 sec
InfoGraph Editor - Installation Package 1 mo 9 days - 1 mo 3 days - #13 57 sec
RTISA - WB-1.1.2 - Build + Deploy (DEV) 4 mo 18 days - #22 N/A 1 min 15 sec
SAMT - trunk 1 hr 44 min - #132 N/A 3 min 3 sec
Static 19 hr - #19 N/A 7 sec
TIGEAR - Trunk 4 hr 24 min - #718 6 days 18 hr - #710 35 sec
VPS - Trunk 16 hr - #191 23 hr - #190 36 sec
VPS Fortify Scan 1 day 15 hr - #58 N/A 3 hr 2 min
Icon: S M L
Legend RSS for all RSS for failures RSS for just latest builds

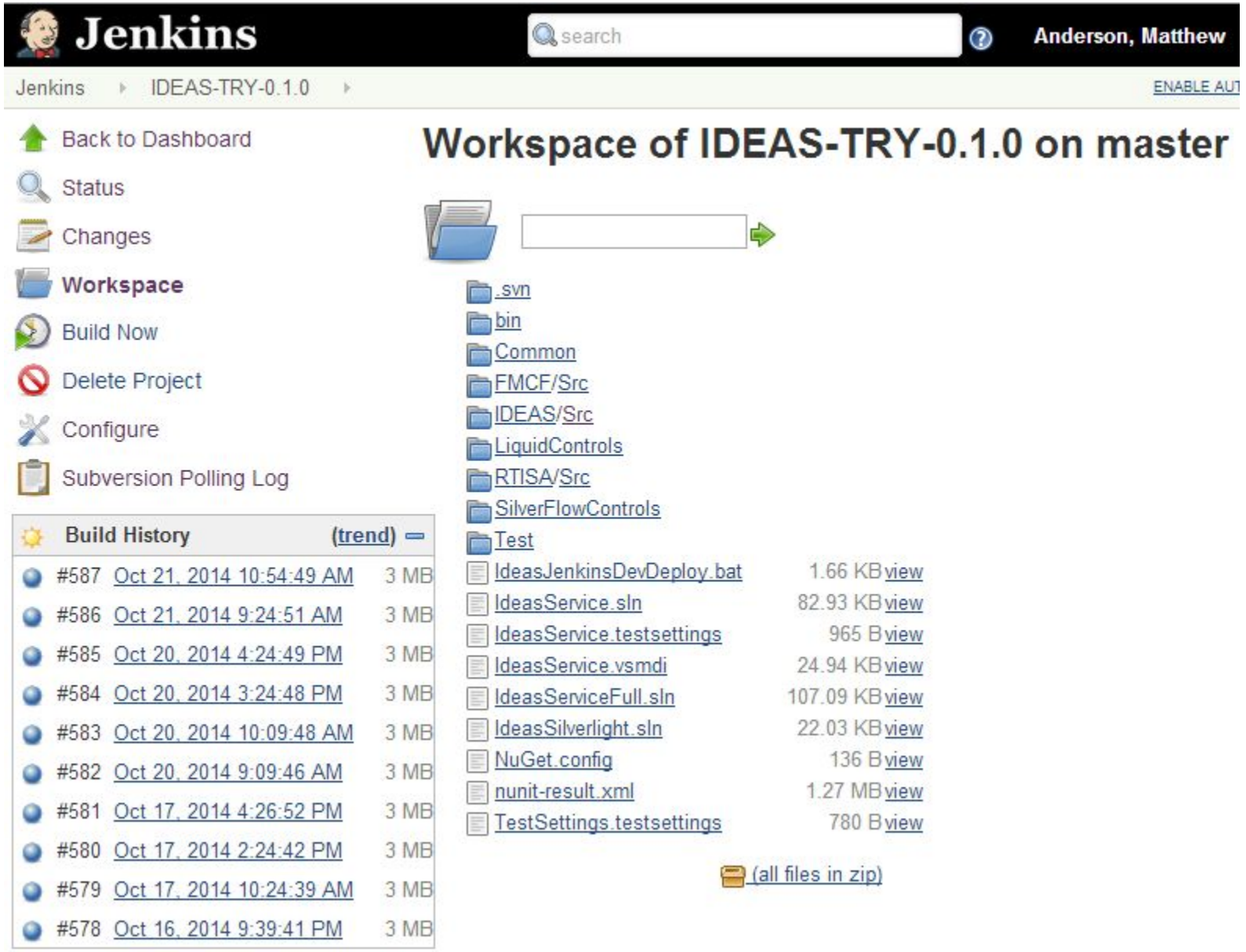
PROJECT VIEW



PROJECT VIEW

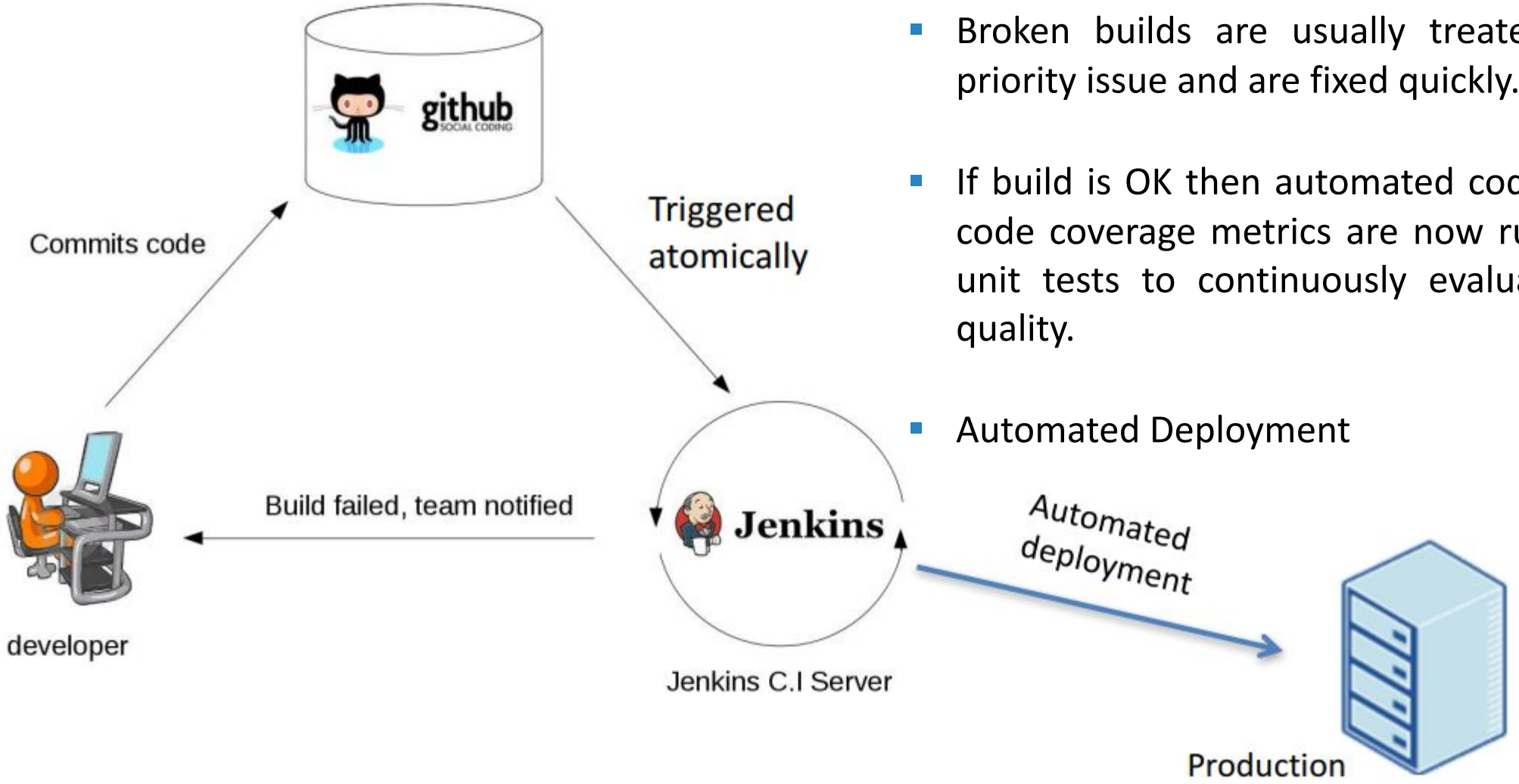


WORKSPACE


Jenkins
search
Anderson, Matthew
Jenkins
IDEAS-TRY-0.1.0
ENABLE AUT
Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Subversion Polling Log
Build History (trend)
#587 Oct 21, 2014 10:54:49 AM 3 MB
#586 Oct 21, 2014 9:24:51 AM 3 MB
#585 Oct 20, 2014 4:24:49 PM 3 MB
#584 Oct 20, 2014 3:24:48 PM 3 MB
#583 Oct 20, 2014 10:09:48 AM 3 MB
#582 Oct 20, 2014 9:09:46 AM 3 MB
#581 Oct 17, 2014 4:26:52 PM 3 MB
#580 Oct 17, 2014 2:24:42 PM 3 MB
#579 Oct 17, 2014 10:24:39 AM 3 MB
#578 Oct 16, 2014 9:39:41 PM 3 MB
Workspace of IDEAS-TRY-0.1.0 on master
.svn
bin
Common
FMCF/Src
IDEAS/Src
LiquidControls
RTISA/Src
SilverFlowControls
Test
IdeasJenkinsDevDeploy.bat 1.66 KB view
IdeasService.sln 82.93 KB view
IdeasService.testsettings 965 B view
IdeasService.vsmdi 24.94 KB view
IdeasServiceFull.sln 107.09 KB view
IdeasSilverlight.sln 22.03 KB view
NuGet.config 136 B view
nunit-result.xml 1.27 MB view
TestSettings.testsettings 780 B view
(all files in zip)

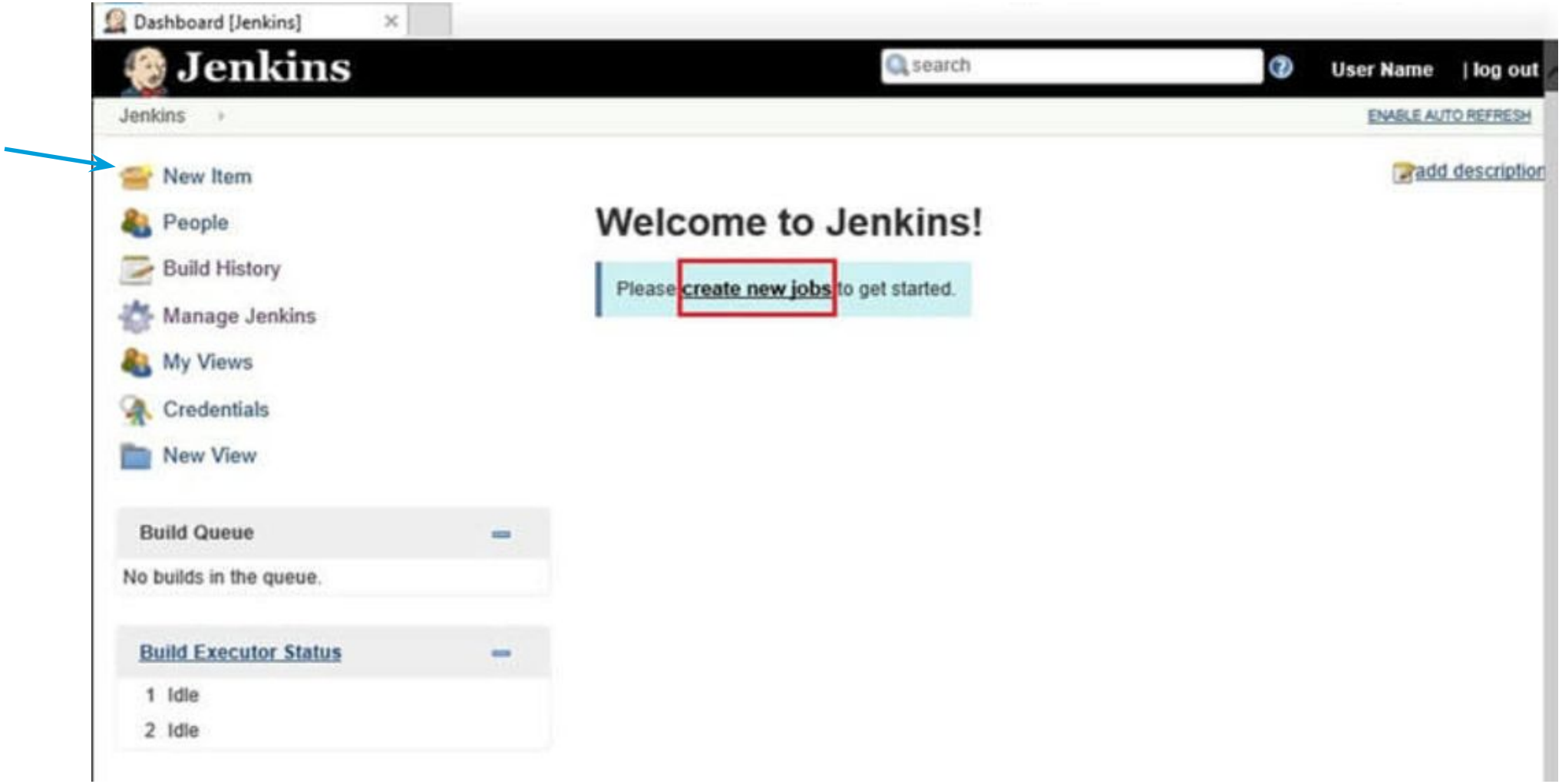
JOB/pipeline Creation

-

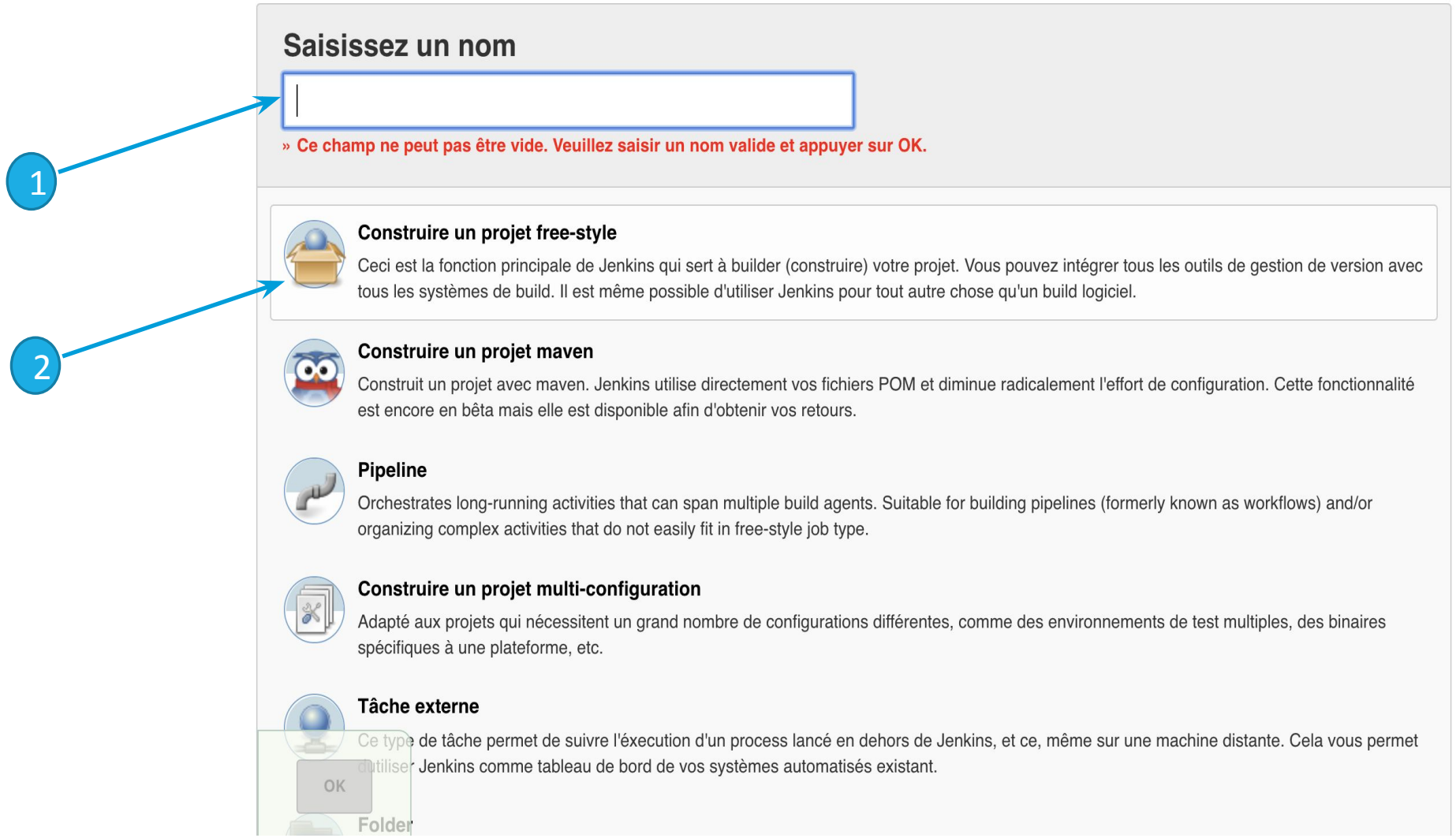


- A build start based on the trigger configuration
- Broken builds are usually treated as a high priority issue and are fixed quickly.
- If build is OK then automated code quality and code coverage metrics are now run along with unit tests to continuously evaluate the code quality.
- Automated Deployment

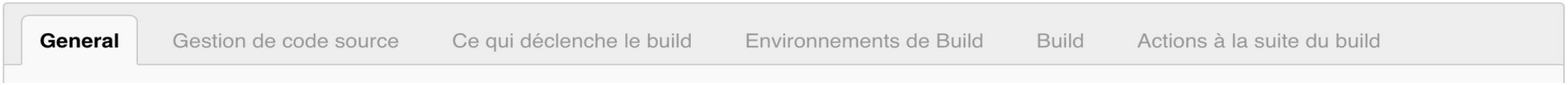
JOB Creation



JOB Creation



JOB



Source Code importation

which SCM ?

Trigger build

- SCM Remote trigger
- Following another build
- Periodically build
- Periodically check on SCM

Build envirenment

- Environment variables
- Workspace cleaning rules
- Configure output results

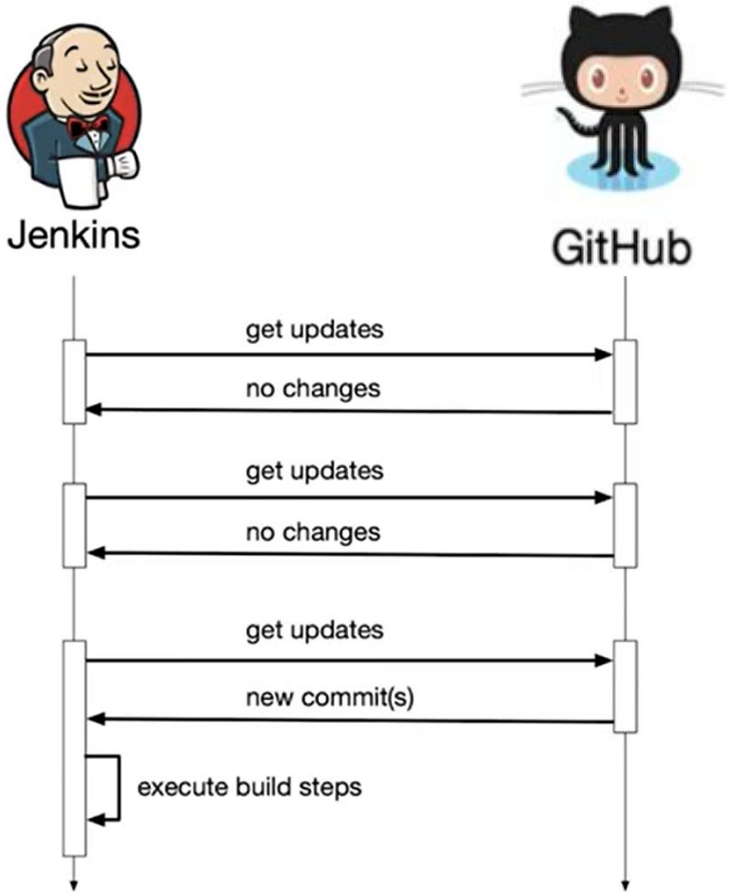
Build Scripts

- Scripts Maven, gradle, ant
- Scripts shell, Ansible, Python..

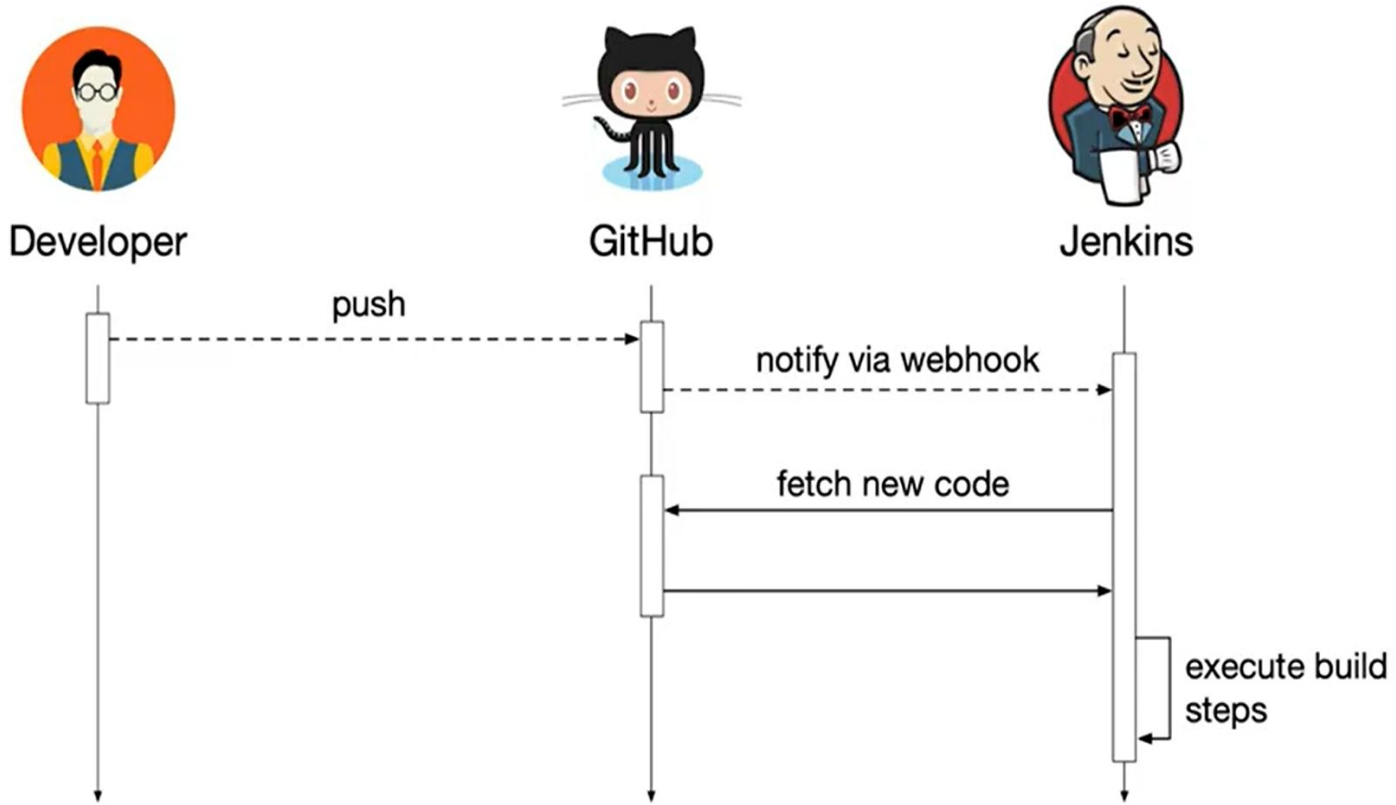
Post-build actions

- Exporting repports
- Send of notifications (Email, Slack..)
- Deployment actions

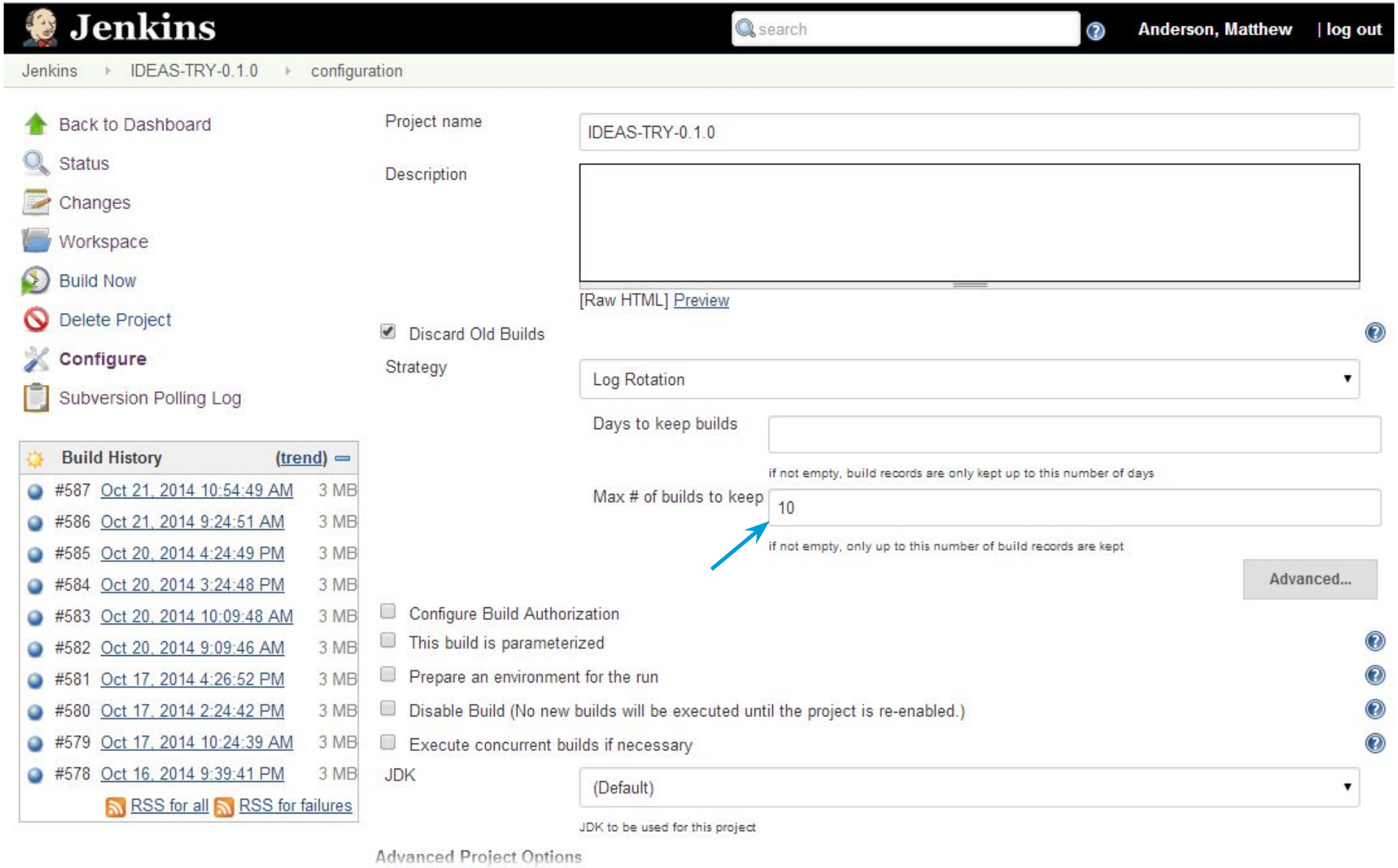
JOB Trigger

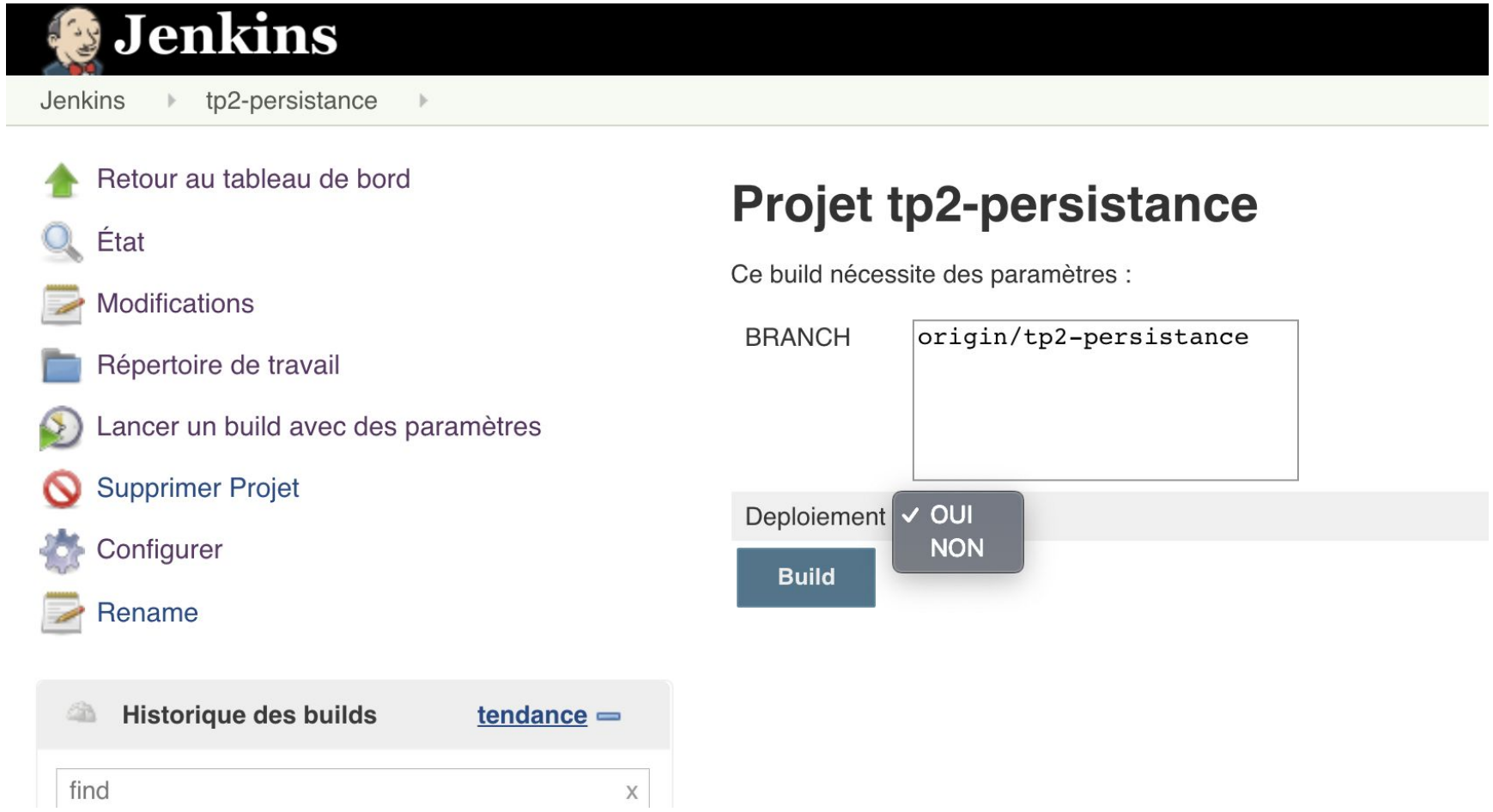


VS



General


Jenkins
search
Anderson, Matthew
| log out
Jenkins
IDEAS-TRY-0.1.0
configuration
Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Subversion Polling Log
Build History
(trend)
#587 Oct 21, 2014 10:54:49 AM 3 MB
#586 Oct 21, 2014 9:24:51 AM 3 MB
#585 Oct 20, 2014 4:24:49 PM 3 MB
#584 Oct 20, 2014 3:24:48 PM 3 MB
#583 Oct 20, 2014 10:09:48 AM 3 MB
#582 Oct 20, 2014 9:09:46 AM 3 MB
#581 Oct 17, 2014 4:26:52 PM 3 MB
#580 Oct 17, 2014 2:24:42 PM 3 MB
#579 Oct 17, 2014 10:24:39 AM 3 MB
#578 Oct 16, 2014 9:39:41 PM 3 MB
RSS for all
RSS for failures
Project name
IDEAS-TRY-0.1.0
Description
[Raw HTML] Preview
Discard Old Builds
Strategy
Log Rotation
Days to keep builds
if not empty, build records are only kept up to this number of days
Max # of builds to keep
10
if not empty, only up to this number of build records are kept
Advanced...
Configure Build Authorization
This build is parameterized
Prepare an environment for the run
Disable Build (No new builds will be executed until the project is re-enabled.)
Execute concurrent builds if necessary
JDK
(Default)
JDK to be used for this project
Advanced Project Options



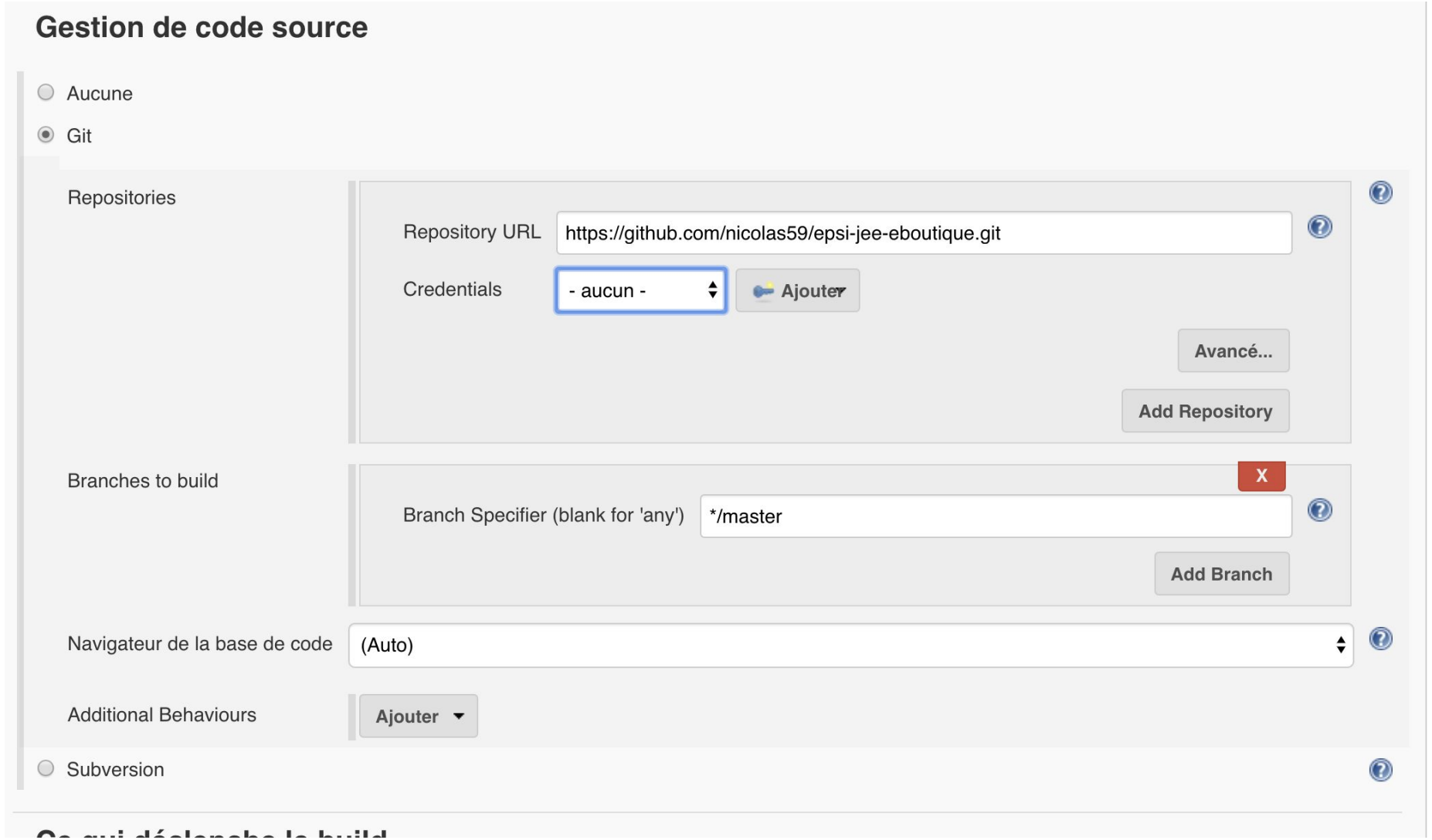
- Nombreuses extensions

Ex : Récupération des branches / tags Git / SVN

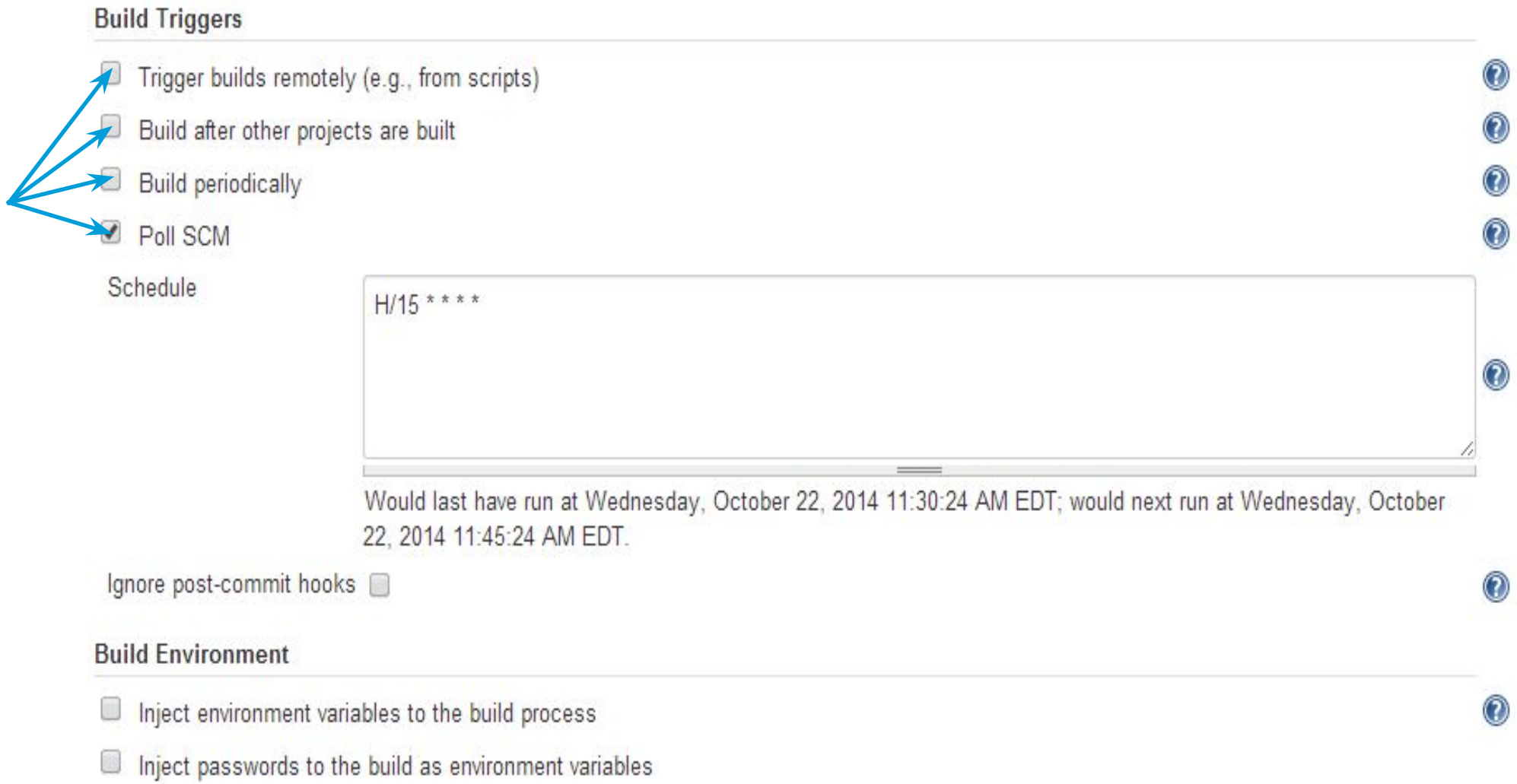
- Définition de liste de valeur, checkbox, saisie manuelle

- Paramètres exploitable dans les scripts de build

Source Code Management



Build Triggers



For all scheduling tasks, Jenkins uses a cron-style syntax, consisting of five fields separated by white space in the following format:

MINUTE HOUR DOM MONTH DOW
with the following values possible for each field:

MINUTE
Minutes within the hour (0–59)

HOUR
The hour of the day (0–23)

DOM
The day of the month (1–31)

MONTH
The month (1–12)

DOW
The day of the week (0–7) where 0 and 7 are Sunday.

There are also a few short-cuts:

“*” represents all possible values for a field. For example, “* * * * *” means “once a minute.”

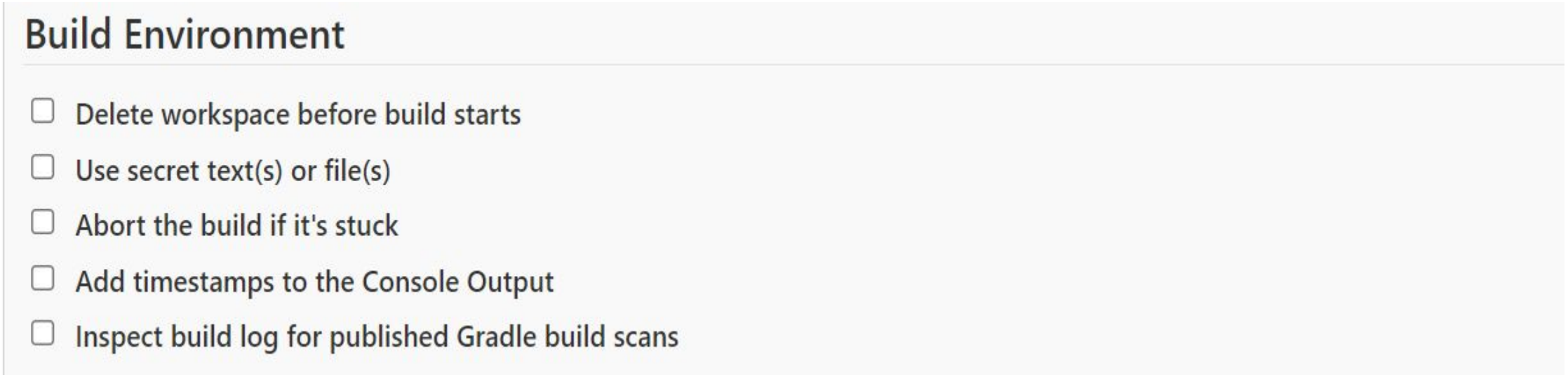
You can define ranges using the “M–N” notation. For example “1-5” in the DOW field would mean “Monday to Friday.”

You can use the slash notation to defined skips through a range. For example, “*/5” in the MINUTE field would mean “every five minutes.”

A comma-separated list indicates a list of valid values. For example, “15,45” in the MINUTE field would mean “at 15 and 45 minutes past every hour.”

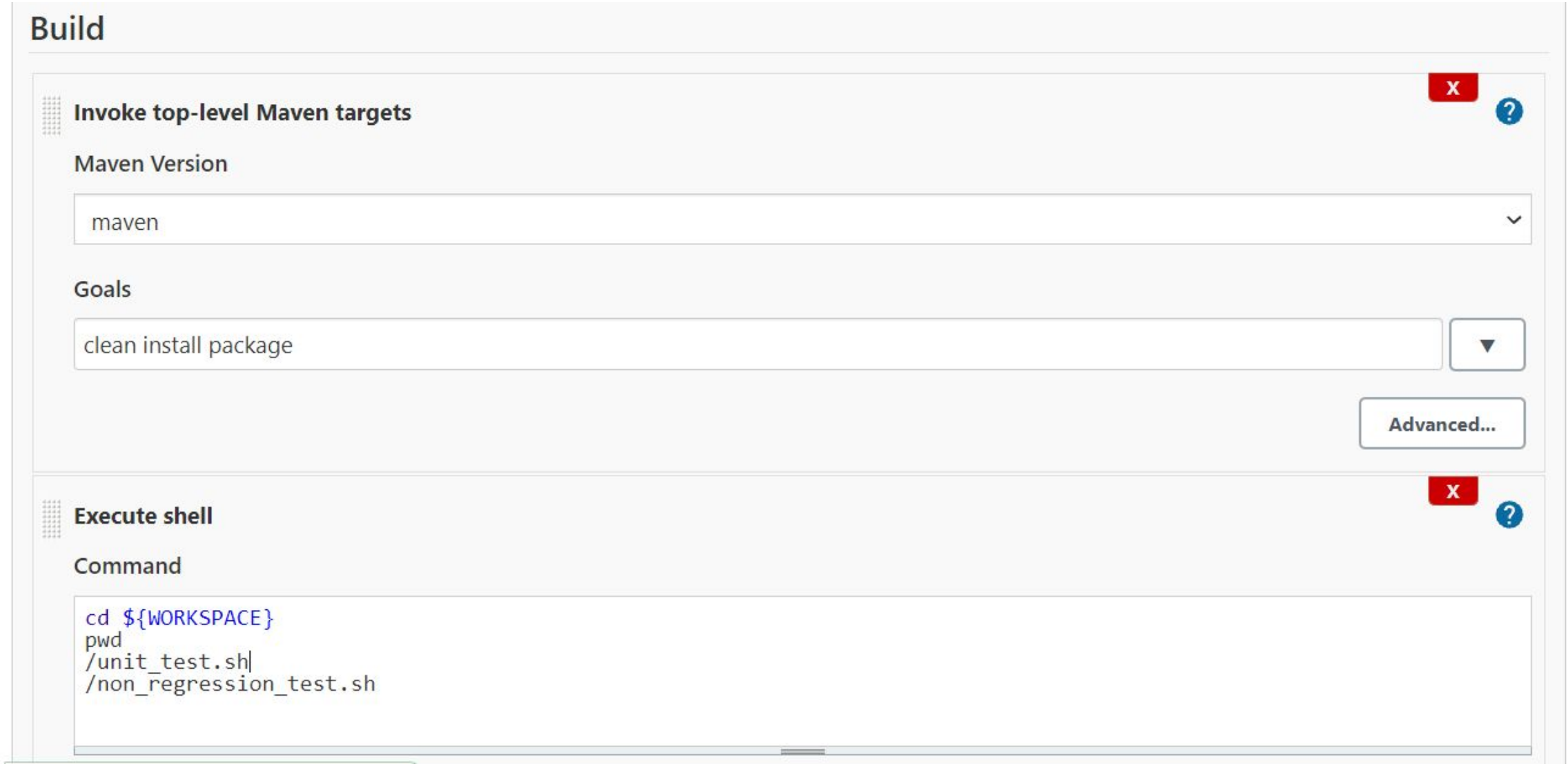
You can also use the shorthand values of “@yearly”, “@annually”, “@monthly”, “@weekly”, “@daily”, “@midnight”, and “@hourly”

Build Environment

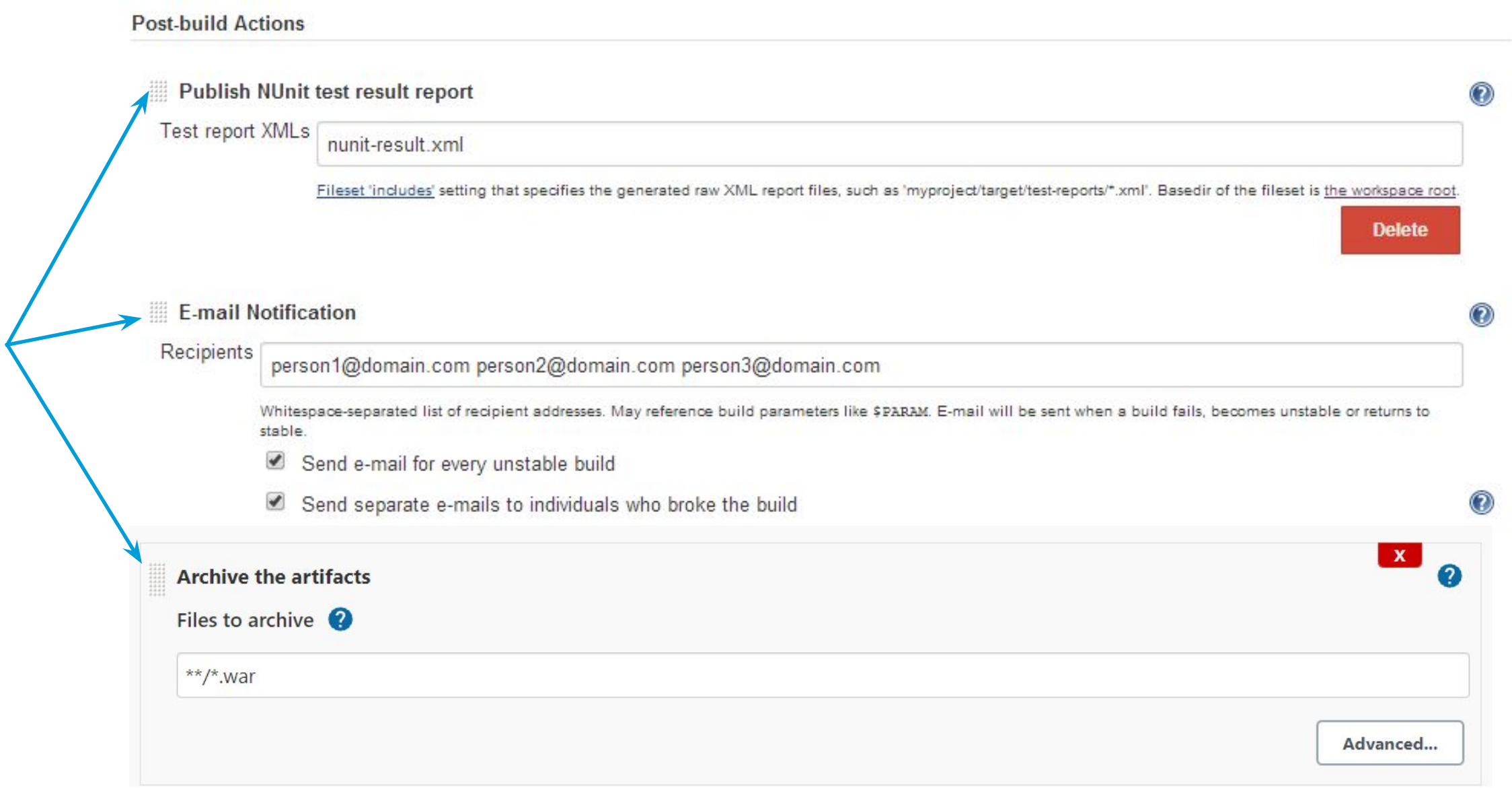


- **Allow the injection of environment variable into this particular job**
- **Workspace cleaning rules**
- **Configure output results**
- **Error handling**

Build


Build
Invoke top-level Maven targets
Maven Version
maven
Goals
clean install package
Advanced...
Execute shell
Command
cd ${WORKSPACE}
pwd
/unit_test.sh
/non_regression_test.sh

Post-Build actions

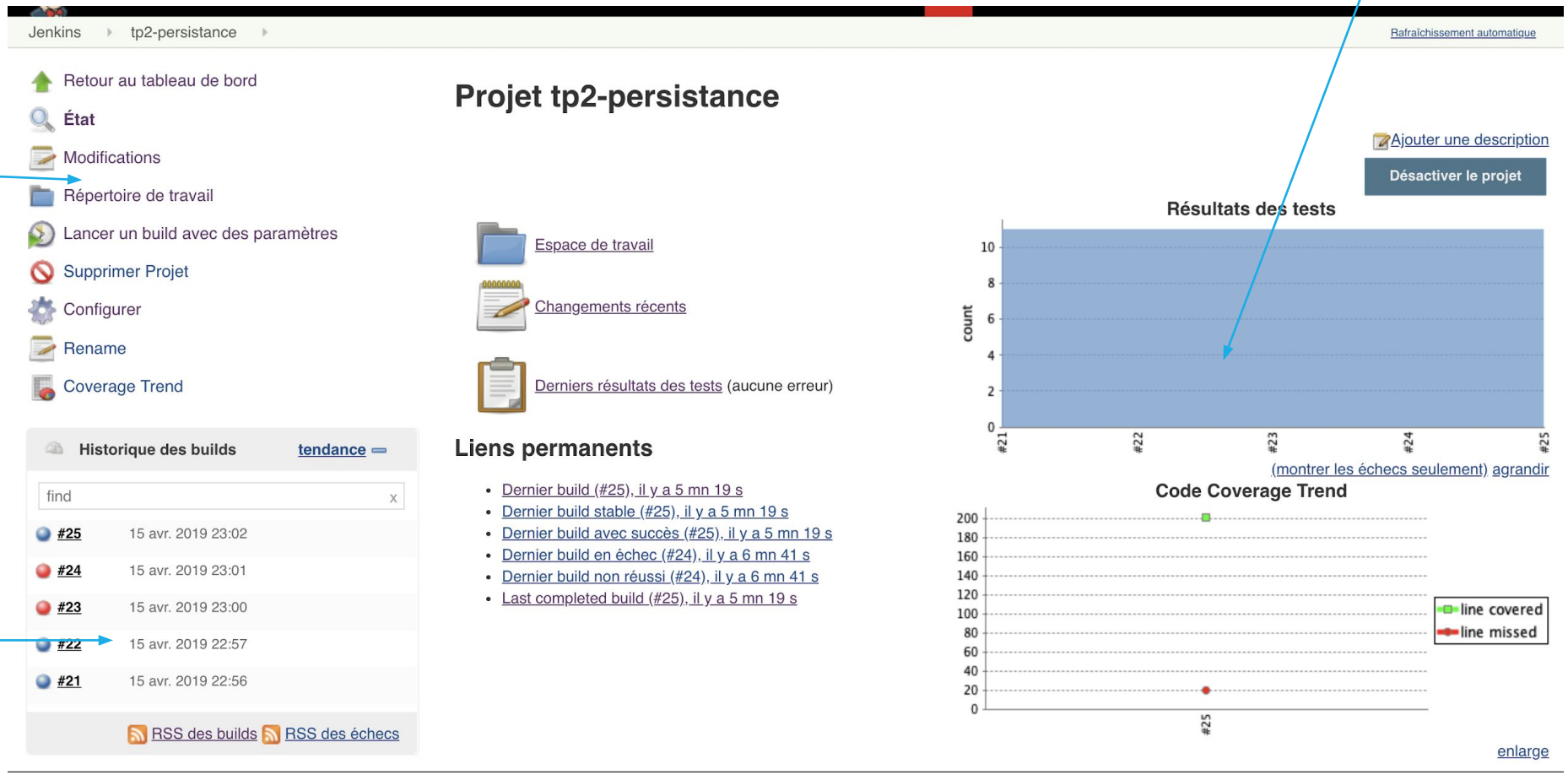


Result Dashboard

TU / Coverage

Access Configuration / Modification

Builds history



Workspace of tp2-persistance on maître

.git			
.settings			
src			
target			
.classpath	15 avr. 2019 22:15:42	1,44 KB	vue
.gitignore	15 avr. 2019 22:15:42	9 B	vue
.project	15 avr. 2019 22:15:42	544 B	vue
pom.xml	15 avr. 2019 22:12:39	2,23 KB	vue
README.md	8 avr. 2019 22:08:08	365 B	vue

(Tous les fichiers dans un zip)

- Possibility to see the workspace

Tip: it is better to remove it in the Post Build phase

=> May take up a lot of space
(Example : node_modules > 400 Mo per build)

Projet tp2-persistance

Ajouter une description

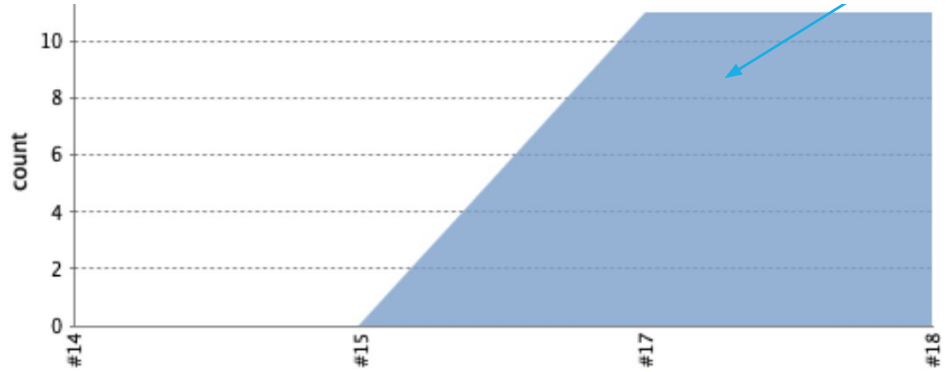
Désactiver le projet

Progress of the passage of TU

Espace de travail

Changements récents

Derniers résultats des tests (aucune erreur)



(montrer les échecs seulement) agrandir

Liens permanents

- Dernier build (#18), il y a 37 s
- Dernier build stable (#18), il y a 37 s
- Dernier build avec succès (#18), il y a 37 s
- Dernier build en échec (#16), il y a 12 mn
- Dernier build non réussi (#16), il y a 12 mn
- Last completed build (#18), il y a 37 s

Status of TU by packages / by TU

Résultats des tests

0 échecs

11 tests

A pris 4 s.

Ajouter une description

Tous les tests

Package	Durée	Échec (diff)	Sauté (diff)	Pass (diff)	Total (diff)
fr.epsi.tp.persistance.dao	4 s	0	0	11 +11	11 +11

Retour au tableau de bord
État
Modifications
Répertoire de travail
Lancer un build
Supprimer Projet
Configurer
Rename

Historique des builds tendance

find x

#17 15 avr. 2019 22:24

Modifications

#15 (15 avr. 2019 22:15:41)

1. Modification gestion exception suite chgt version java — nicolas.rousseau1 / githubweb

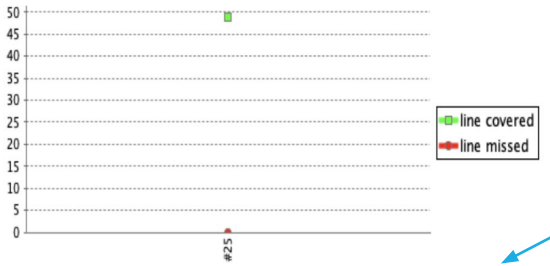
#14 (15 avr. 2019 22:12:38)

1. Modification version java — nicolas.rousseau1 / githubweb

```
Démarré par l'utilisateur nicolas
Running as SYSTEM
Building in workspace /Users/nrousseau1/.jenkins/workspace/tp2-persistance
using credential Github
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/nicolas59/epsi-correction.git # timeout=10
Fetching upstream changes from https://github.com/nicolas59/epsi-correction.git
 > git --version # timeout=10
using GIT_ASKPASS to set credentials
 > git fetch --tags --force --progress https://github.com/nicolas59/epsi-correction.git +refs/heads/*:refs/remotes/origin/*
 > git rev-parse refs/remotes/origin/tp2-persistance^{commit} # timeout=10
 > git rev-parse refs/remotes/origin/origin/tp2-persistance^{commit} # timeout=10
Checking out Revision e1fef17f33e11e8a5a33d6778f8ba13729e07ae2 (refs/remotes/origin/tp2-persistance)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f e1fef17f33e11e8a5a33d6778f8ba13729e07ae2
Commit message: "Modification niveau de log"
 > git rev-list --no-walk 113200c34c8cc73271dfb29ea163435b926882cb # timeout=10
[tp2-persistance] $ /Users/nrousseau1/Documents/01.Logiciels/01.Dev/01-JAVA/apache-maven-3.5.4/bin/mvn clean install test
[INFO] Scanning for projects...
[INFO]
[INFO] -----------------< fr.epsi.tp:epsi-jee-persistance-jdbc >-----------------
[INFO] Building epsi-jee-persistance-jdbc 0.0.1-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ epsi-jee-persistance-jdbc ---
[INFO] Deleting /Users/nrousseau1/.jenkins/workspace/tp2-persistance/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ epsi-jee-persistance-jdbc ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ epsi-jee-persistance-jdbc ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 10 source files to /Users/nrousseau1/.jenkins/workspace/tp2-persistance/target/classes
[WARNING] /Users/nrousseau1/.jenkins/workspace/tp2-persistance/src/main/java/fr/epsi/tp/persistance/dao/IJdbcCrud.java: /Users/nrousseau1/.jenkins/workspace/tp2-
persistance/src/main/java/fr/epsi/tp/persistance/dao/IJdbcCrud.java uses unchecked or unsafe operations.
[WARNING] /Users/nrousseau1/.jenkins/workspace/tp2-persistance/src/main/java/fr/epsi/tp/persistance/dao/IJdbcCrud.java: Recompile with -Xlint:unchecked for details.
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ epsi-jee-persistance-jdbc ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 2 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ epsi-jee-persistance-jdbc ---
```

Allows you to easily determine problems during compilation and correct

Package: CommandeDAO



Statistiques par classes

CommandeDAO

name	instruction	branch	complexity	lin
CommandeDAO()	M: 0 C: 8 100%	M: 0 C: 0 100%	M: 0 C: 1 100%	M: 0 C: 2 100%
create(Commande)	M: 0 C: 91 100%	M: 4 C: 6 60%	M: 4 C: 2 33%	M: 0 C: 24 100%
findById(Long)	M: 0 C: 88 100%	M: 2 C: 4 67%	M: 2 C: 2 50%	M: 0 C: 23 100%

Coverage

Couverture par ligne

```
15: import fr.epsi.tp.persistance.bean.CommandeLigne;
16:
17: public class CommandeDAO implements IJdbcCrud<Commande, Long> {
18:
19:   private ProduitDAO produitDAO = new ProduitDAO();
20:
21:   public Commande findById(Long identifier) throws SQLException {
22:     ResultSet rs = null;
23:     try (Connection conn = ConnectionFactory.getInstance()
24:       .getConnection();
25:         PreparedStatement ps = conn.prepareStatement("select id, date_creation, produit_
26:             + "inner join comm_produit on comm_produit.commande_id=commande.id where id=
27:       ps.setLong(1, identifier);
28:       rs = ps.executeQuery();
29:       Commande commande = null;
30:•      if (rs.next()) {
31:         commande = new Commande();
32:         commande.setIdentifier(rs.getLong(1));
33:         Timestamp ts = rs.getTimestamp("date_creation");
34:         commande.setDateCreation(ts.toInstant().atZone(ZoneId.of("UTC"))
35:         .toLocalDate());
36:
37:
38:         List<CommandeLigne> comLines = new ArrayList<>();
```

Jobs

LABS:

- lab1_jenkins.txt
- lab2_jenkins.txt
- lab3_jenkins.txt
- lab4_jenkins.txt

Pipeline

LABS:

- jenkins_file_first_pipeline_demo.txt
- jenkins_file_for_maven_pipeline.txt
- jenkins_file_for_nodejs_pipeline.txt

Distributed jenkins (master/slave)

LABS:

- create_agents_lab.txt
- agent_job.txt
- agent_pipeline.txt

SonarQube

LABS:

- sonar installation.txt

SonaQube Integration for code Quality & Security

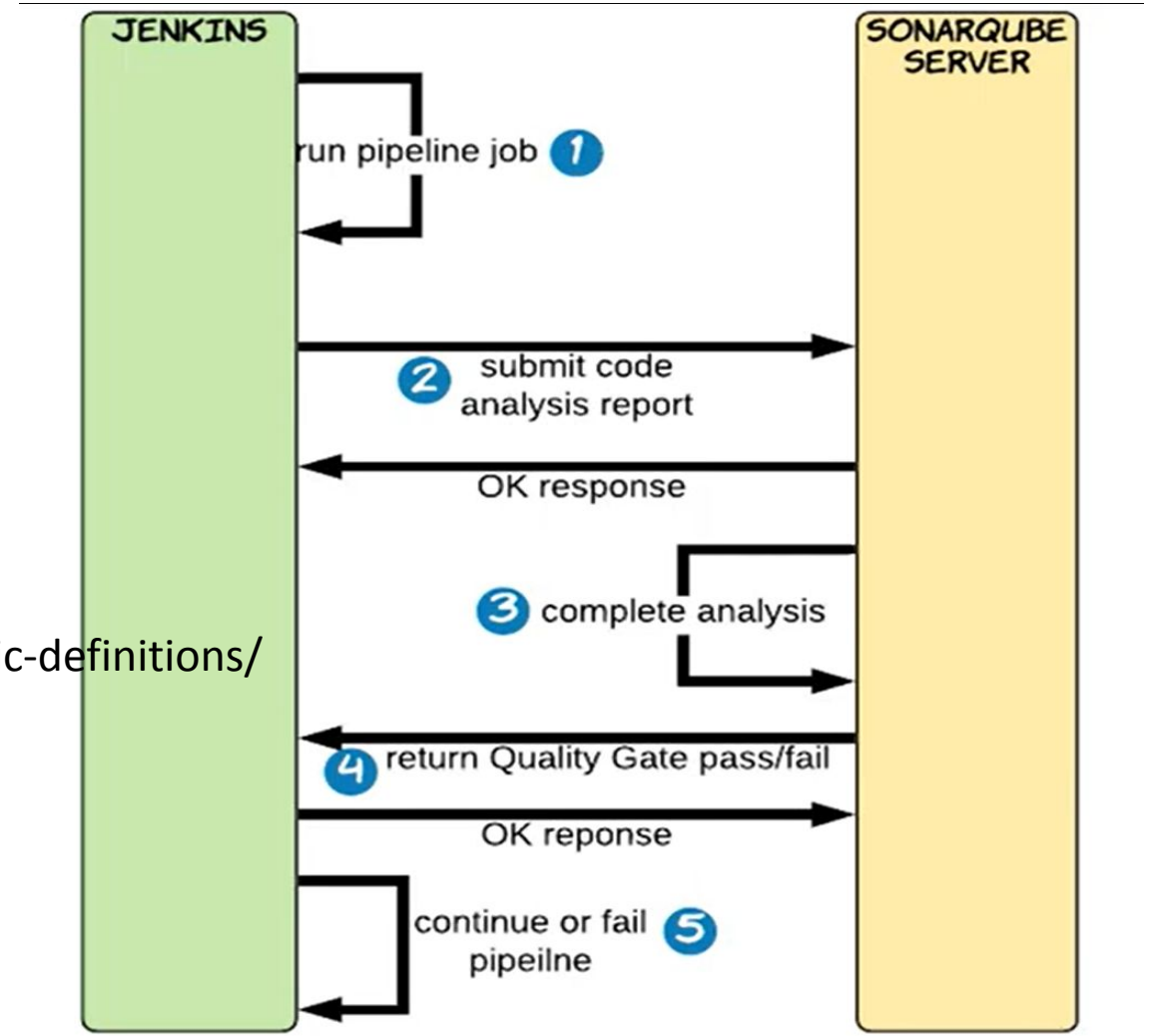
LAB: jenkins_pipeline_SonarQube.txt

```
<build>
  <pluginManagement>
    <plugins>
      <plugin>
        <groupId>org.sonarsource.scanner.maven</groupId>
        <artifactId>sonar-maven-plugin</artifactId>
        <version>3.7.0.1746</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>
```

https://docs.sonarqube.org/latest/user-guide/metric-definitions/

SonaQube Gating

https://docs.sonarqube.org/latest/user-guide/metric-definitions/



SonaQube Gating (gradle example)

```
plugins {
    id 'org.sonarqube' version '3.3'
    // any other plugins
}
```

We also need to include a configuration to tell the SonarQube scanner where to find the SonarQube server that we have running:

```
sonarqube {
    properties {
        property 'sonar.host.url', 'http://localhost:9000'
    }
}
```

Lastly, to ensure the Jacoco test report is always created when we run the *sonarqube* task let's setup the following `dependsOn` relationship:

```
tasks.named('sonarqube').configure {
    dependsOn test
}
```

LAB: jenkins_pipeline_sonar_Quality_Gate.txt

Parametrized and Multibranching Pipeline

LAB: jenkins_pipeline_Parametrized_branching.txt

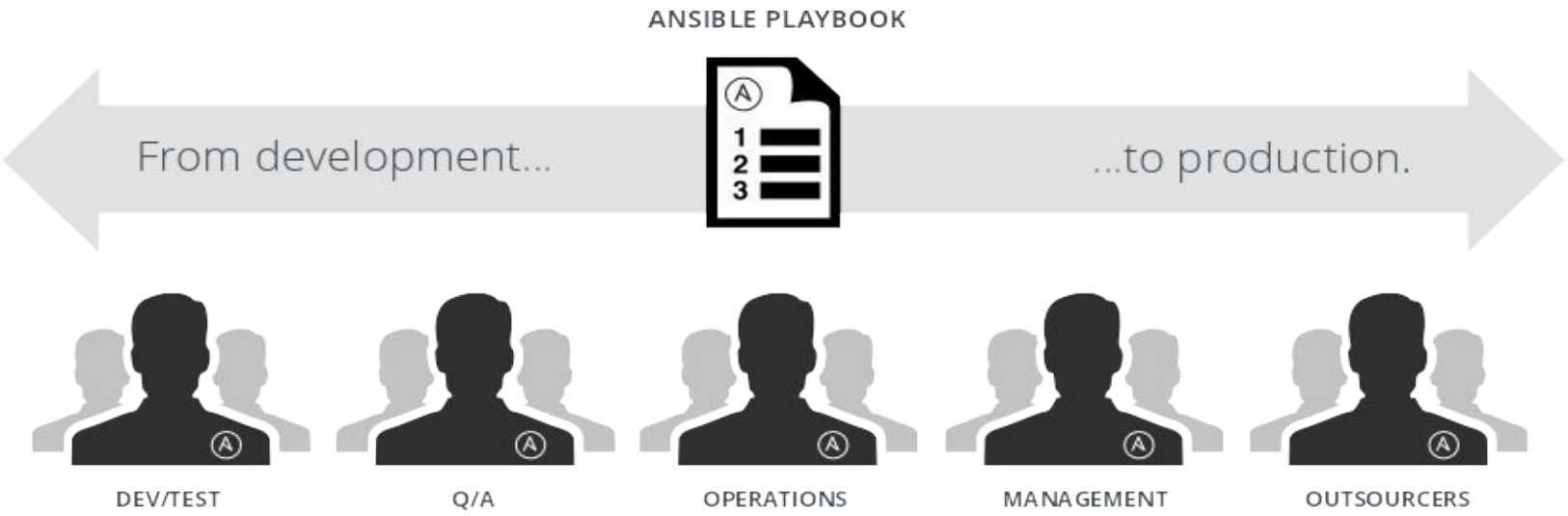
LAB: jenkins_pipeline_Multibranching_automatic_execution.txt

ANSIBLE
by Red Hat®

Ansible Hands-on Introduction

What is Ansible?

It's a simple automation language that can perfectly describe an IT application infrastructure in Ansible Playbooks.



COMMUNICATION IS THE KEY TO DEVOPS.

Ansible is the first **automation language** that can be read and written across IT.

Ansible is the only **automation engine** that can automate the entire **application lifecycle** and **continuous delivery** pipeline.

Why Ansible ?

Is the latest entrant in the market compared to Puppet, Chef which:

- **Difficult for new users who must have advanced skills in Ruby programming language**
- **Use multiple masters which complicates the management process**
- **Need agent**

Why Ansible ?

SIMPLE

Human readable automation

No special coding skills needed

Tasks executed in order

Get productive quickly

POWERFUL

App deployment

Configuration management

Workflow orchestration

Orchestrate the app lifecycle

AGENTLESS

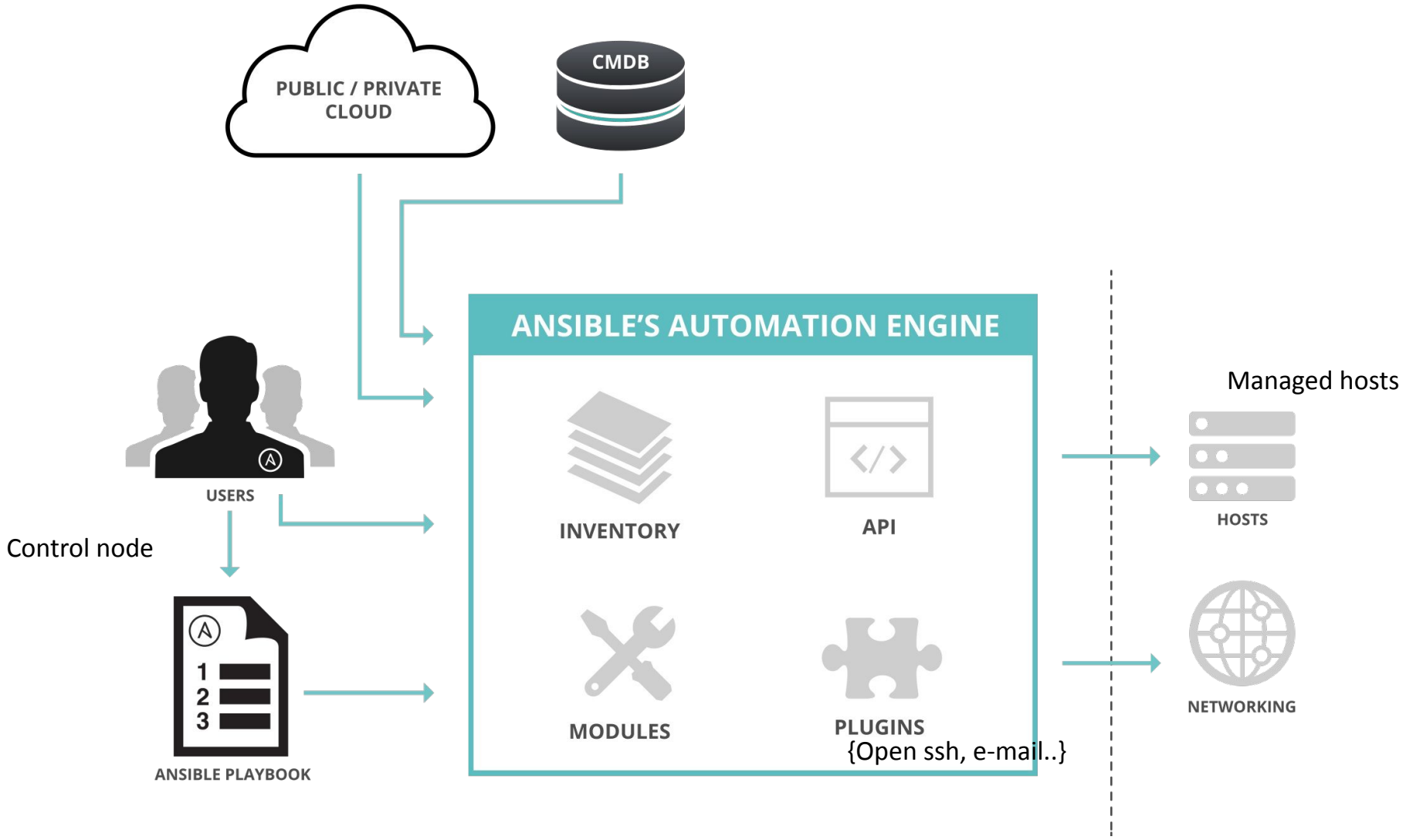
Agentless architecture

Uses OpenSSH & WinRM

No agents to exploit or update

More efficient & more secure

How Ansible Works



Inventory

Inventory is a collection of hosts that Ansible can connect and manage.

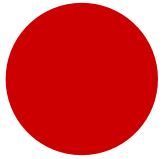
INI-style file that contain list of host names, ip adress, groups

```
10.42.0.2
10.42.0.6
10.42.0.7
[qualif-server]
10.42.0.8
10.42.0.100
host.example.co
m
```

Ansible Configuration File

The behavior of an ansible installation can be cusomized by modifying setting in the ansible.cfg

/etc/ansible/ansible.cfg
~/.ansible.cfg
./ansible.cfg



```
inventory = /etc/ansible/hosts
...
[privilege_escalation]
become=True
become_method=sudo
become_user=root
become_ask_pass=False
```

Modules

Modules are bits of code transferred to the target system and executed to satisfy the task declaration. Ansible ships with several hundred today!

- YUM / Apt
- COPY
- FILE
- Get_url
- Git
- Ping
- Debug
- Service
- Synchronize
- Template
- Cron

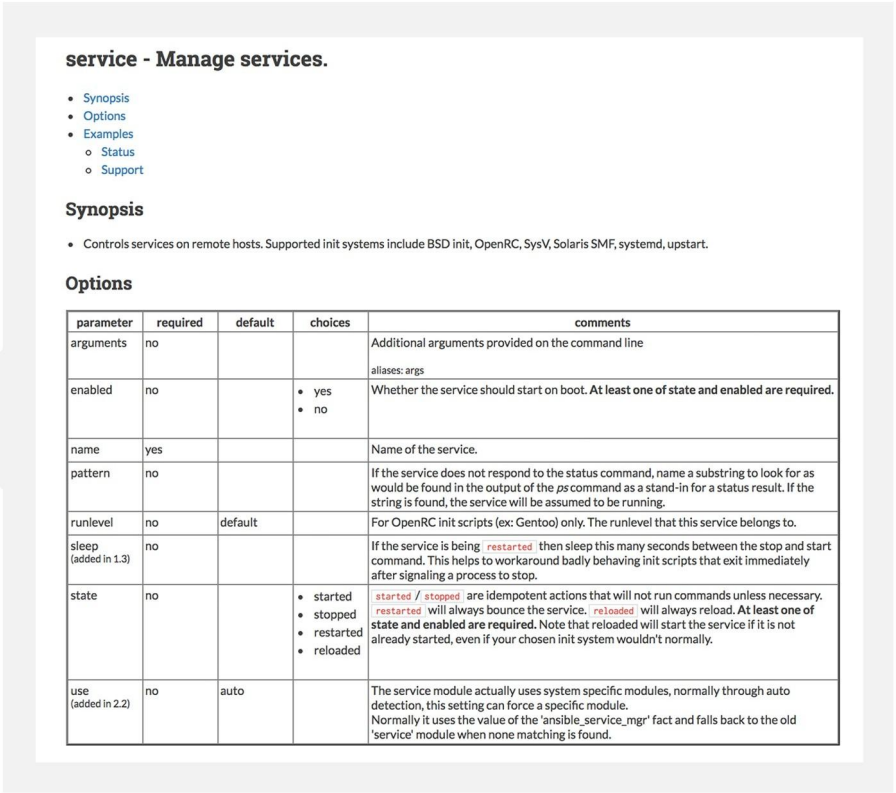
Modules Documentation

http://docs.ansible.com/

Docs » Module Index

Module Index

- All Modules
- Cloud Modules
- Clustering Modules
- Commands Modules
- Crypto Modules
- Database Modules
- Files Modules
- Identity Modules
- Inventory Modules
- Messaging Modules
- Monitoring Modules
- Network Modules
- Notification Modules
- Packaging Modules
- Remote Management Modules
- Source Control Modules
- Storage Modules
- System Modules
- Utilities Modules
- Web Infrastructure Modules
- Windows Modules



service - Manage services.

- Synopsis
- Options
- Examples
  - Status
  - Support

Synopsis

- Controls services on remote hosts. Supported init systems include BSD init, OpenRC, SysV, Solaris SMF, systemd, upstart.

Options

parameter	required	default	choices	comments
arguments	no			Additional arguments provided on the command line aliases: args
enabled	no		• yes • no	Whether the service should start on boot. **At least one of state and enabled are required.**
name	yes			Name of the service.
pattern	no			If the service does not respond to the status command, name a substring to look for as would be found in the output of the *ps* command as a stand-in for a status result. If the string is found, the service will be assumed to be running.
runlevel	no	default		For OpenRC init scripts (ex: Gentoo) only. The runlevel that this service belongs to.
sleep (added in 1.3)	no			If the service is being `restarted` then sleep this many seconds between the stop and start command. This helps to workaround badly behaving init scripts that exit immediately after signaling a process to stop.
state	no		• started • stopped • restarted • reloaded	`started` / `stopped` are idempotent actions that will not run commands unless necessary. `restarted` will always bounce the service. `reloaded` will always reload. **At least one of state and enabled are required.** Note that reloaded will start the service if it is not already started, even if your chosen init system wouldn't normally.
use (added in 2.2)	no	auto		The service module actually uses system specific modules, normally through auto detection, this setting can force a specific module. Normally it uses the value of the 'ansible_service_mgr' fact and falls back to the old 'service' module when none matching is found.

Modules Documentation

```
# List out all modules installed
$ ansible-doc -l
...
cop
y
cron
...

# Read documentation for installed module
$ ansible-doc copy
> COPY

  The [copy] module copies a file on the local box to remote locations. Use the [fetch] module to copy files from remote locations
  to the local  box. If you need variable interpolation in copied files, use the [template] module.
  * note: This module has a corresponding action

plugin.  Options (= is mandatory):
```

Modules: Run Commands

If Ansible doesn't have a module that suits your needs there are the "run command" modules:

command: Takes the command and executes it on the host. The most secure and predictable.

script: Runs a local script on a remote node after transferring it.

Ad-Hoc Commands

An ad-hoc command is a single Ansible task to perform quickly, but don't want to save for later.

```
# check all my inventory hosts are ready to be managed by Ansible
$ ansible all -m ping

# collect and display the discovered facts # for the localhost
$ ansible localhost -m setup

# run the uptime command on all hosts in the # web group
$ ansible web -m command -a "touch /home/centos/file.txt"

# insert a content into file with specific user for the production group server
$ ansible production -m copy -a 'content="Welcome to server Managed by Ansible\n" dest=/etc/motd'
-u devops

# install the apache server on the qualification group
$ ansible qualification -m yum -a 'name=httpd state=present enabled=yes' -b
```

Plays & Playbooks

Plays are ordered sets of tasks to be executed on managed hosts. A playbook is a YAML file containing one or more plays.

Easy to write, human readable format

Example

```
---
- name: install and start
  apache  hosts: web
  become:
  yes  vars:
    http_port: 80

  tasks:
   -name: httpd package is
    present  yum:
      name:
      httpd
      state:
      latest

   -name: latest index.html file is
    present  copy:
      src:
      files/index.html
      dest:
      /var/www/html/
```

- ansible-playbook play.yml -i hosts --syntax-check
- ansible-playbook play.yml -i hosts -check
- ansible-playbook play.yml -i hosts
- ansible-playbook play.yml -i hosts –step
- ansible-playbook play.yml -i hosts -vvv

Plays & Playbooks

Lab1:
http://www.devops-training-levelup.com/ansible-training/labs/Lab1_first_playbook.txt

Lab2:
http://www.devops-training-levelup.com/ansible-training/labs/Lab2_second_playbook.txt

Doing More Than a simple Playbook

Here are some more essential playbook features that you can apply:

- **Variables**
- **Loops**
- **Templates**
- **Conditionals**
- **Block/always/ rescue**
- **Notify/ handlers**
- **Roles**

Variables

Ansible can work with multiple source of variables

- **Variables in playbook**
- **Host variable / group variables**
- **Registered variables**
- **Gathered facts**

```
---
- name: Install Apache and start the service
hosts: production, serverc.lab.example.com
vars:
web_pkg: httpd
firewall_pkg: firewalld
web_service: httpd
firewall_service: firewalld
python_pkg: python-httplib2
rule: http
tasks:
      - name: Install the required packages
        yum:
          name:
            - "{{ web_pkg }}"
            - "{{ firewall_pkg }}"
            - "{{ python_pkg }}"
          state: latest
      - name: Start and enable the {{ firewall_service }}
      service
        service:
          name: "{{ firewall_service }}"
          enabled: true
          state: started
      - name: Start and enable the {{ web_service }}
      service
          service:
            name: "{{ web_service }}"
            enabled: true
```

```
      - name: Create web content to be
      served
        copy:
          content: "Example web content"
          dest: /var/www/html/index.html
      - name: Open the port for {{ rule }}
        firewalld:
            service: "{{ rule }}"
            permanent: true
            immediate: true
            state: enabled
- name: Verify the Apache service
hosts: localhost
tasks:
      - name: Ensure the webserver is
      reachable
        uri:
            url: http://servera.lab.example.com
            status_code: 200
        register: output
      - name: Debug the previous task
        debug:
          var: output
```

vars_files to replace the **vars:**

Best Practice:
group_vars/production.yml {rule: https}
Host_vars//serverc.lab.example.com.yml {rule:http}

Playbook variables

Lab 3:
http://www.devops-training-levelup.com/ansible-training/labs/Lab3_playbook_with_variables.txt

Lab 4:
http://www.devops-training-levelup.com/ansible-training/labs/Lab4_playbook_with_registred_variables.txt

Gathering Facts

Facts are bits of information derived from examining a host systems that are stored as variables for later use in a play.

```
$ ansible localhost -m setup  localhost | success
>> {  "ansible_facts": {
     "ansible_default_ipv4": {  "address":
        "192.168.1.37",
        "alias": "wlan0",  "gateway": "192.168.1.1",
        "interface": "wlan0",
        "macaddress": "c4:85:08:3b:a9:16",  "mtu": 1500,
        "netmask": "255.255.255.0",
        "network": "192.168.1.0",
        "type": "ether"
     },
     …
      "ansible_nodename": "qfoielb01a.novalocal",
           "ansible_os_family": "RedHat",
           "ansible_pkg_mgr": "yum",
           "ansible_processor": [
              "GenuineIntel",
              "Intel(R) Xeon(R) Gold 6161 CPU @ 2.20GHz"
           ],
           "ansible_processor_cores": 1,
           "ansible_processor_count": 1,
```

Playbook variables

Lab 5:
http://www.devops-training-levelup.com/ansible-training/labs/Lab5_playbook_with_captured_facts.txt

Loops

Is a list of items that Ansible reads and iterate over:

defined by providing a list of items to the **loop**

```
- yum:
    name: "{{ item }}"
    state: latest

    loop:
    - httpd
    - mod_wsgi
```

Playbook loop

Lab 6:
http://www.devops-training-levelup.com/ansible-training/labs/Lab6_playbook_with_simple_loop.txt

Conditionals

Ansible supports the conditional execution of a task

```
- yum:
    name: httpd
    state: latest
  when: httpd is defined
```

Playbook Conditional

Lab 7:
http://www.devops-training-levelup.com/ansible-training/labs/Lab7_playbook_with_conditional.txt

```
---
- hosts: all
vars:
  db_package: mariadb-server
  db_service: mariadb
  db_users:
      - db_admin
      - db_user
   configure_database_path: /etc/my.cnf
tasks:
       - name: Create  2 users for MariaDB
         user:
           name: "{{ item }}"
         with_items: "{{ db_users }}"
         when: inventory_hostname in groups['databases']
       - name: Install packages on redhat distribution servers
         yum:
            name: "{{ item }} ”
            state: latest
         with_item:
         - mariadb-server
         - httpd
            when: db_package is defined and ansible_facts.distribution ==
      "RedHat"
      - name: Install packages on debian distribution servers
         apt:
            name: "{{ item }} ”
         with_item:
         - mariadb-server
         - httpd
            when: db_package is defined and ansible_facts.distribution
      ==   ”Debian"
```

Playbook Combining loop with conditional

Lab 8:

http://www.devops-training-levelup.com/ansible-training/labs/Lab8_Combining_conditional_with_loop.txt

Playbook

Lab 9:
http://www.devops-training-levelup.com/ansible-training/labs/Lab9_playbook_Mariadb_full_deployment.txt

Notify/ handlers

Handlers are special tasks that run at the end of a play if notified by another task when a change occurs.

If a task that includes a notify does not execute (for example, a package is already installed),
the handler will not be notified. The handler will be skipped unless another task notifies it.

Example Handler Task in a Play

```
tasks:
- name: mysql package is present
  yum:
   name: mysql
   state: latest
  notify: restart mysql

  - name: template vhost file
    template:
      src: vhost.conf.j2
      dest: /etc/httpd/conf.d/vhost.conf
      owner: root
      group: root
      mode: 0644
    notify:
       - restart httpd
  '''
  '''

  handlers:
    -name: restart httpd
      service:
        name: httpd
        state: restarted
   -name: restart mysql
      service:
        name: mysql
        state: restarted
```

```
TASK [myvhost : template vhost file]
*****************************************

changed: [servera.lab.example.com]
RUNNING HANDLER [myvhost : restart httpd]
*****************************************

changed: [servera.lab.example.com]
PLAY RECAP
*************************************************************

servera.lab.example.com : ok=8 changed=5 unreachable=0 failed=0
```

Playbook Notify - Handler

Lab 10:

http://www.devops-training-levelup.com/ansible-training/labs/Lab10_playbook_Notify_Handler.txt

Ansible blocks and error handling

Blocks allow for logical grouping of tasks and used to control how tasks are executed and handle errors in combination with the **rescue** and always statements: If any task in a block fails, tasks in its rescue block are executed in order to recover

always: Defines the tasks that will always run independently of the success or failure of tasks

```
- block:
  -shell:
     cmd: /usr/local/lib/create-database-collection
  -shell:
     cmd: /usr/local/lib/upgrade-database
  rescue:
  -shell:
     cmd: /usr/local/lib/database-rollback
  always:
  -Service:
     name: mariadb
     State: restarted
```

redhat.

Playbook Block-resuce-always

Lab 11:

Transform the conditional_playbook.yml of the Lab7 using the block/rescue/always statements

http://www.devops-training-levelup.com/ansible-training/materials/block-rescue-always_playbook.yml

Error

```
---
- hosts: mailservers
vars:
maildir_path: /home/john/Maildir
maildir: /home/student/Maildir
mail_package: postfix
mail_service: postfix
tasks:
- name: Create {{ maildir_path }}
  copy:
    src: "{{ maildir }}"
    dest: "{{ maildir_path }}"
    mode: 0755
- name: Install {{ mail_package }} package
  yum:
     name: "{{ mail_package }}"
     state: latest
```

```
[student@workstation demo-failures]$ ansible-playbook playbook.yml
... Output omitted ...
TASK [Create /home/john/Maildir]
*********************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failed": true,
"msg": "could not find src=/home/student/Maildir"}
NO MORE HOSTS LEFT
*************************************************************
to retry, use: --limit @playbook.retry
... Output omitted ...
```

Ansible blocks and error handling

```
---
- hosts: mailservers
vars:
maildir_path: /home/john/Maildir
maildir: /home/student/Maildir
mail_package: postfix
mail_service: postfix
tasks:
- name: Create {{ maildir_path }}
  copy:
    src: "{{ maildir }}"
    dest: "{{ maildir_path }}"
    mode: 0755
  ignore_errors: yes
- name: Install {{ mail_package }} package
  yum:
    name: "{{ mail_package }}"
    state: latest
```

```
[student@workstation demo-failures]$ ansible-playbook.playbook.yml
... Output omitted ...
TASK [Create /home/john/Maildir]
*********************************************
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failed": true,
"msg": "could not find src=/home/student/Maildir"}
...ignoring
TASK [Install postfix package]
*********************************************
changed: [servera.lab.example.com]
.... Output omitted ...
```

Templates

Super powerful module used to put a files onto a remote host using jinja2 templating language

It works similarly to the copy module, but with 2 major differences:

- **template looks for templates in ./templates/ when you supply a relative path for src (instead of ./files/ for copy)**
- **You can use the jinja2 templating language in your files, which will be templated out separately for each remote host**

```
├── ansible.cfg
├── templates
│   └── my_app.conf.j2
└── playbook.yml
```

With ./templates/my_app.conf.j2 containing:

```
local_ip = {{ ansible_default_ipv4["address"] }}
local_user = {{ ansible_user }}
```

Running the following task in playbook.yml:

```
- name: install my_app configuration file from template
  template:
    src: my_app.conf.j2
    dest: $HOME/my_app.conf
```

Produces the following on my Ubuntu 18.04 test host:

```
local_ip = 10.1.11.72
local_user = ubuntu
```

And the following on my Centos 7.5 test host:

```
local_ip = 10.1.11.62
local_user = centos
```

```
This is the system {{ ansible_hostname }}.
Last deployment date is: {{ ansible_date_time.date }}.
Only use this system with permission.
You can ask {{ system_owner }} for access.
```

```
---
- hosts: all
user: devops
become: true
vars:
system_owner: devops@example.com
tasks:
- template:
    src: motd.j2
    dest: /etc/motd
    owner: root
    group: root
    mode: 0644
```

```
[student@workstation jinja2]$ ssh
devops@servera.lab.example.com
This is the system servera.
 Last deployment date is: 2020-04-12.
Only use this system with permission.
You can ask devops@example.com for access.
[devops@servera ~]# exit
Connection to servera.lab.example.com closed.
```

```
[student@workstation jinja2]$ ssh
devops@serverb.lab.example.com
This is the system serverb.
Last deployment date is: 2020-04-12.
Only use this system with permission.
You can ask devops@example.com for access.
[devops@serverb ~]# exit
Connection to servera.lab.example.com closed.
```

First Playbook Template

Lab 12:

http://www.devops-training-levelup.com/ansible-training/labs/Lab12_playbook_template.txt

Other example: dynamic /etc/hosts file

Used to transpose an FQDN machine name (domain name) to an IP address, even before querying the DNS servers. Used for local area networks (LAN) and virtual networks (VLAN)

```
{% for variable in groups['all'] %}
{{hostvars[variable].ansible_default_ipv4.address}} {{hostvars[variable].ansible_fqdn}} {{hostvars[variable].ansible_hostname}}
{% endfor %}
```

```
---
- hosts: all
  tasks:
    - template:
        src: exemple2.j2
        dest: /etc/hosts
        owner: root
        group: root
        mode: 0644
```

```
# Ansible managed
# loopback managed by orange.hosts
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

# hosts lists managed by orange.hosts
172.17.0.189    qfoieinf01a     qfoieinf01a.novalocal
10.228.194.100  qfoieprx01a     qfoieprx01a.novalocal
172.17.0.215    qfoiedb01a      qfoiedb01a.novalocal
172.17.0.203    qfoiedb02b      qfoiedb02b.novalocal
172.17.0.226    qfoiedb03a      qfoiedb03a.novalocal
172.17.0.192    qfoiedb04b      qfoiedb04b.novalocal
10.228.194.101  qfoieprx02b     qfoieprx02b.novalocal
172.17.0.183    qfoiewsc01a     qfoiewsc01a.novalocal
172.17.0.92     qfoiewsc02b     qfoiewsc02b.novalocal
172.17.0.229    qfoiewsc03a     qfoiewsc03a.novalocal
172.17.0.185    qfoieelk01a     qfoieelk01a.novalocal
172.17.0.41     qfoieelk02b     qfoieelk02b.novalocal
10.228.194.102  qfoieelk03a     qfoieelk03a.novalocal
10.228.192.101  qfoiejks01a     qfoiejks01a.novalocal
```

Other example: Load Balancer

```
global
    maxconn 50000
    log /dev/log local0
    user haproxy
    group haproxy
    stats socket /run/haproxy/admin.sock user haproxy group haproxy
Defaults
{% if {{haproxytype}} == ‘happy' %}
    timeout connect 10s
{% else %}
    timeout connect 1s
{% endif %}
    log global
    mode http
    option httplog
frontend www.mysite.com
    bind 10.0.0.3:80
    bind 10.0.0.3:443 ssl crt /etc/ssl/certs/mysite.pem
    default_backend web_servers

backend web_servers
    balance roundrobin
    default-server check maxconn 20
{% for variable in groups[‘backends’]%}
server {{hostvars[variable].ansible_hostname}} {{hostvars[variable].ansible_default_ipv4.address}} :80 cookie
{% endfor %}
```

```
---
- hosts: all
  tasks:
    - template:
        src: loadblancer.j2
        dest=/etc/haproxy/haproxy.cfg
        owner: root
        group: root
        mode: 0644
```

Other example: Load Balancer

```
global
   maxconn 50000
   log /dev/log local0
   user haproxy
   group haproxy
   stats socket /run/haproxy/admin.sock user haproxy
group haproxy
Defaults
   timeout connect 10s
   log global
   mode http
   option httplog
frontend www.mysite.com
   bind 10.0.0.3:80
   bind 10.0.0.3:443 ssl crt /etc/ssl/certs/mysite.pem
   default_backend web_servers

backend web_servers
   balance roundrobin
   default-server check maxconn 20
   server server2 10.0.1.4:80 cookie server2
 server server3 10.0.1.5:80 cookie server3
 server server4 10.0.1.6:80 cookie server4
 server server5 10.0.1.7:80 cookie server5
 server server6 10.0.1.8:80 cookie server6
```

```
global
   maxconn 50000
   log /dev/log local0
   user haproxy
   group haproxy
   stats socket /run/haproxy/admin.sock user haproxy group haproxy
Defaults
   timeout connect 1s
   log global
   mode http
   option httplog
frontend www.mysite.com
   bind 10.0.0.3:80
   bind 10.0.0.3:443 ssl crt /etc/ssl/certs/mysite.pem
   default_backend web_servers

backend web_servers
   balance roundrobin
   default-server check maxconn 20
   server server2 10.0.1.4:80 cookie server2
 server server3 10.0.1.5:80 cookie server3
 server server4 10.0.1.123:80 cookie server4
 server server5 10.0.1.7:80 cookie server5
 server server6 10.0.1.8:80 cookie server6
```

Add/delete server !!! Changement network config !!!

Managing inclusions

When working with complex or long playbooks, administrators can use separate files to divide tasks

```
---
- name: Install fileserver packages
hosts: fileservers
tasks:
  - name: Includes the variable
    include_vars: package.yml
  - name: Installs the package
    include: install_package.yml
```

```
[student@workstation demo-vars-inclusions]$ ansible-playbook
playbook.yml
PLAY [Install fileserver packages]
**************************************

Chapter 4. Managing Variables and Inclusions
146 DO407-A2.0-en-1-20160804
TASK [setup]
**********************************************************

ok: [servera.lab.example.com]
TASK [Includes the variable]
**********************************************

ok: [servera.lab.example.com]
TASK [Installs the package]
**********************************************

included: /home/student/demo-vars-inclusions/install_package.yml
for servera.lab.example.com
TASK [Installs httpd]
**************************************************

changed: [servera.lab.example.com]
PLAY RECAP
******************************************************

servera.lab.example.com : ok=4 changed=1 unreachable=0 failed=0
```

Roles

Roles are a packages of closely related Ansible content that can be shared more easily than plays alone.

- Improves readability and maintainability of complex plays
- Eases sharing, reuse and standardization of automation processes

Roles

An Ansible role's functionality is defined by its directory structure.

```
[user@host roles]$ tree user.example
user.example/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
└── vars
    └── main.yml
```

(contain default values of role variables)

(contains static files that are referenced by role tasks)

(contain the role's handler definitions)

(include author, license, company, and optional role dependencies...)

(contains the role's task definitions)

(contains Jinja2 templates that are referenced by role tasks.)

(Contain the role's variable values.)

redhat.

Project with Embedded Roles Example

```
---
- hosts: remote.example.com
pre_tasks:
- debug:
msg: 'hello'
roles:
- role1
- role2
tasks:
- debug:
msg: 'still busy'
post_tasks:
- debug:
msg: 'goodbye
```

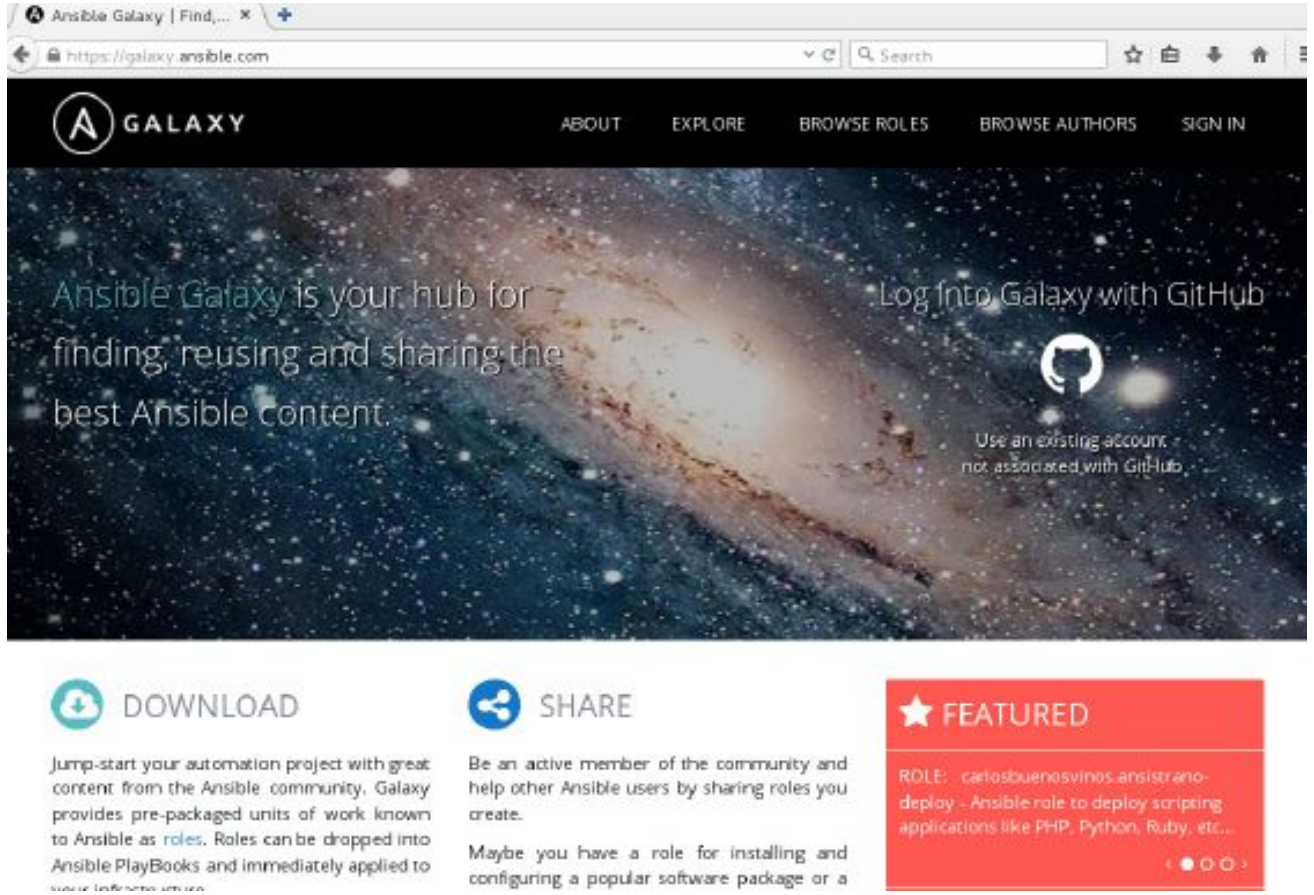
```
mkdir -p roles/role1/{defaults,handlers,tasks}

mkdir -p roles/role2/{defaults,handlers,tasks,vars,files}
```

Ansible Galaxy

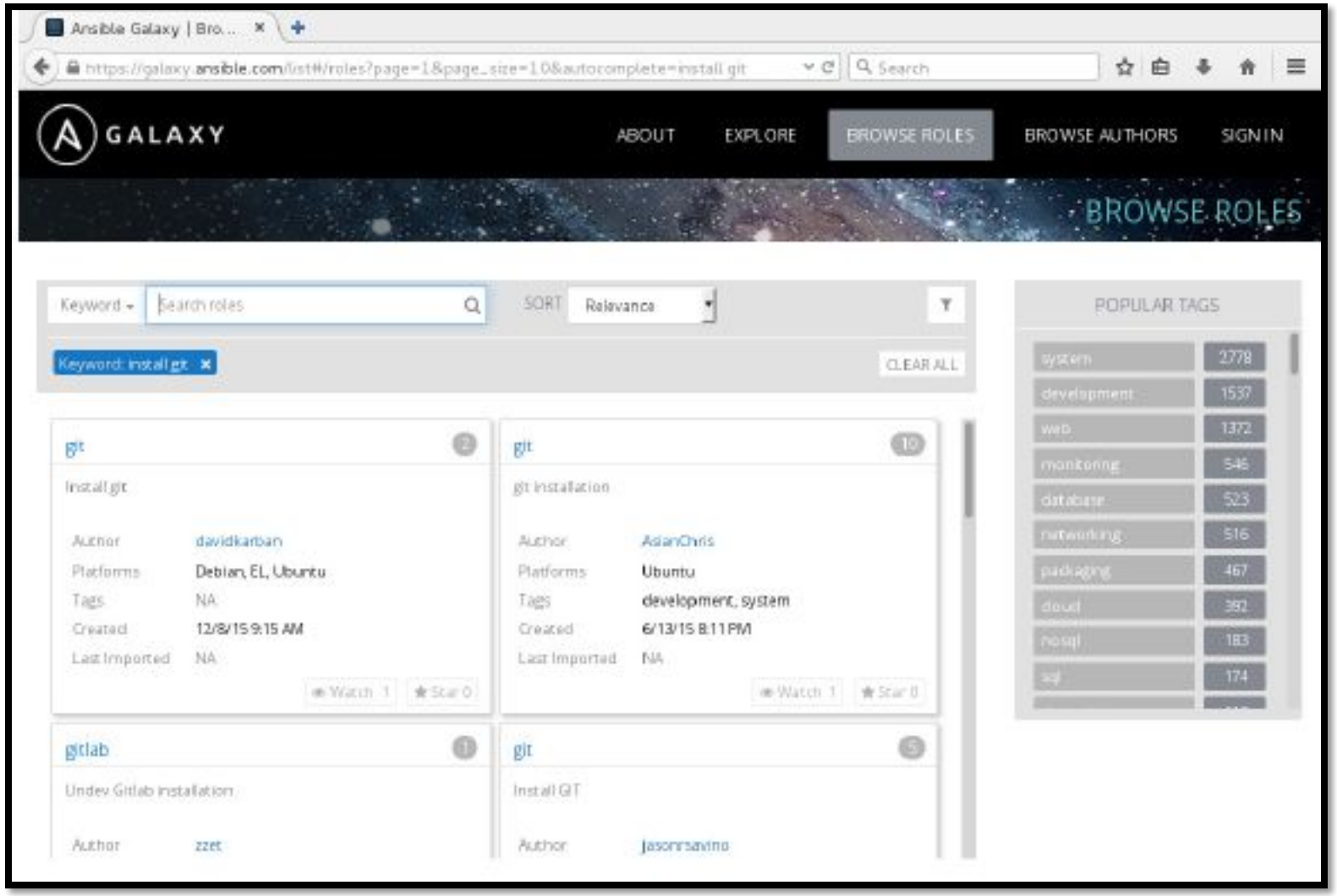
http://galaxy.ansible.com

Ansible Galaxy is a hub for finding, reusing and sharing Ansible content.

Jump-start your automation project with content contributed and reviewed by the Ansible community.



redhat.




```
[user@host ~]$ ansible-galaxy install davidkarban.git -p roles/
- downloading role 'git', owned by davidkarban
- downloading role from https://github.com/davidkarban/ansible-git/archive/master.tar.gz
- extracting davidkarban.git to roles/davidkarban.git
- davidkarban.git was installed successfully
[user@host ~]$ ls roles/
davidkarban.git
```

redhat.

Ansible Galaxy

ansible-galaxy install -r requirements.yml
-r option specifies the requirements file listing the roles to download and install

```
# from GitHub, overriding the name and specifying a specific tag
- src: https://github.com/bennojoy/nginx
  version: master
  name: nginx_role
  scm: git
```

ansible-galaxy init --offline -p roles/role_example

redhat.

Ansible Galaxy

Yum install rhel-system-roles*
roles system distributed by RedHat for multiple use_cases

- kdump
- network
- selinux
- storage
- timesync

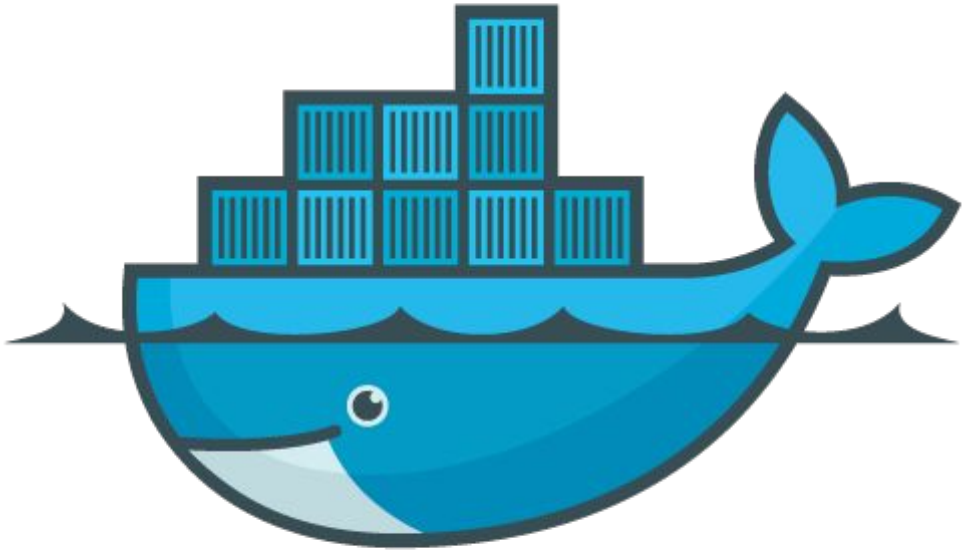
```
/usr/share/ansible/roles/
```

Ansible Vault

Vault:
Ansible may need access to sensitive data such as passwords or API keys in order to configure remote servers.

ansible-vault:

- **Create**
- **Encrypt**
- **Decrypt**
- **Rekey**
- **View**
- **Edit**



docker

www.docker.io

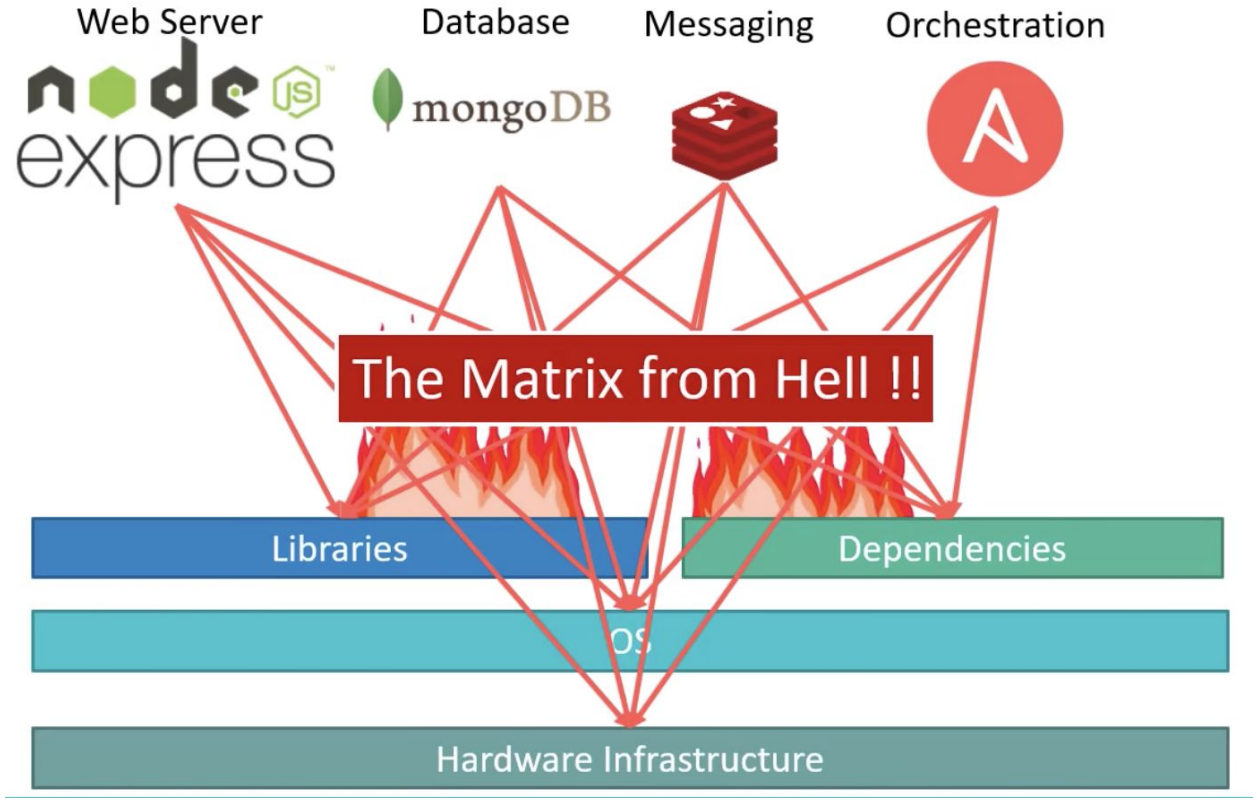
Needs for Containers

- Every application has its own dependencies and libreries
- Every application has it's own specific OS requirements/needs (drivers & binaries files)
- Certain versions of services have issues of compatibility with the OS in case of the application's upgrade
- Issues where one service/tool/component requires one specific version of a dependent OS libreries wheras another service requires another version of the same libreries

docker

Needs for Containers

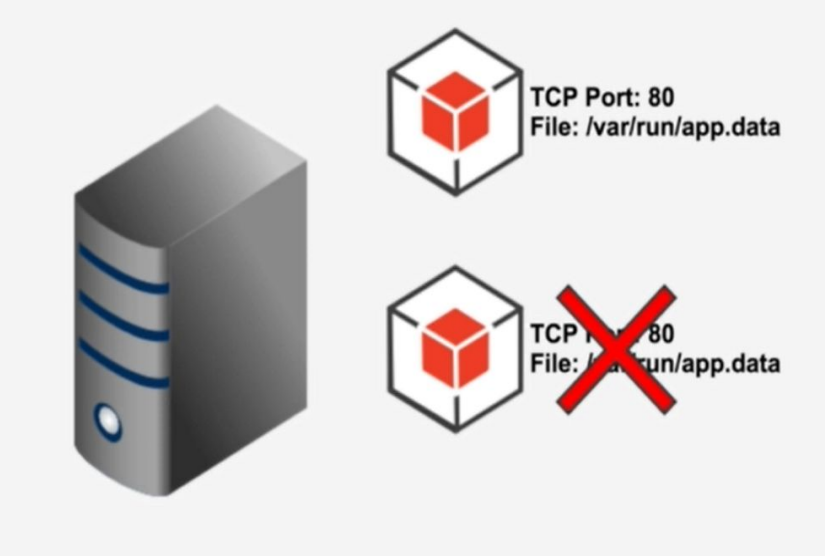
- Every time something changed we had the same issues of checking the compatibilities between the application components versions and the underlying architecture. Which called the « Matrix Of Hell »

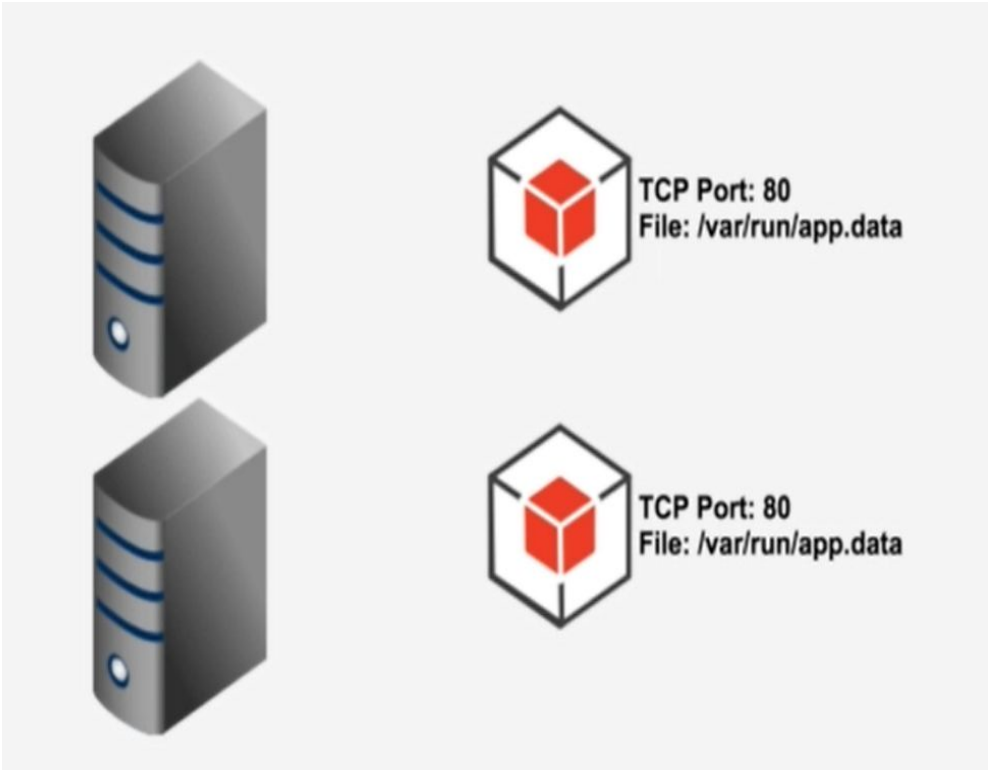


docker

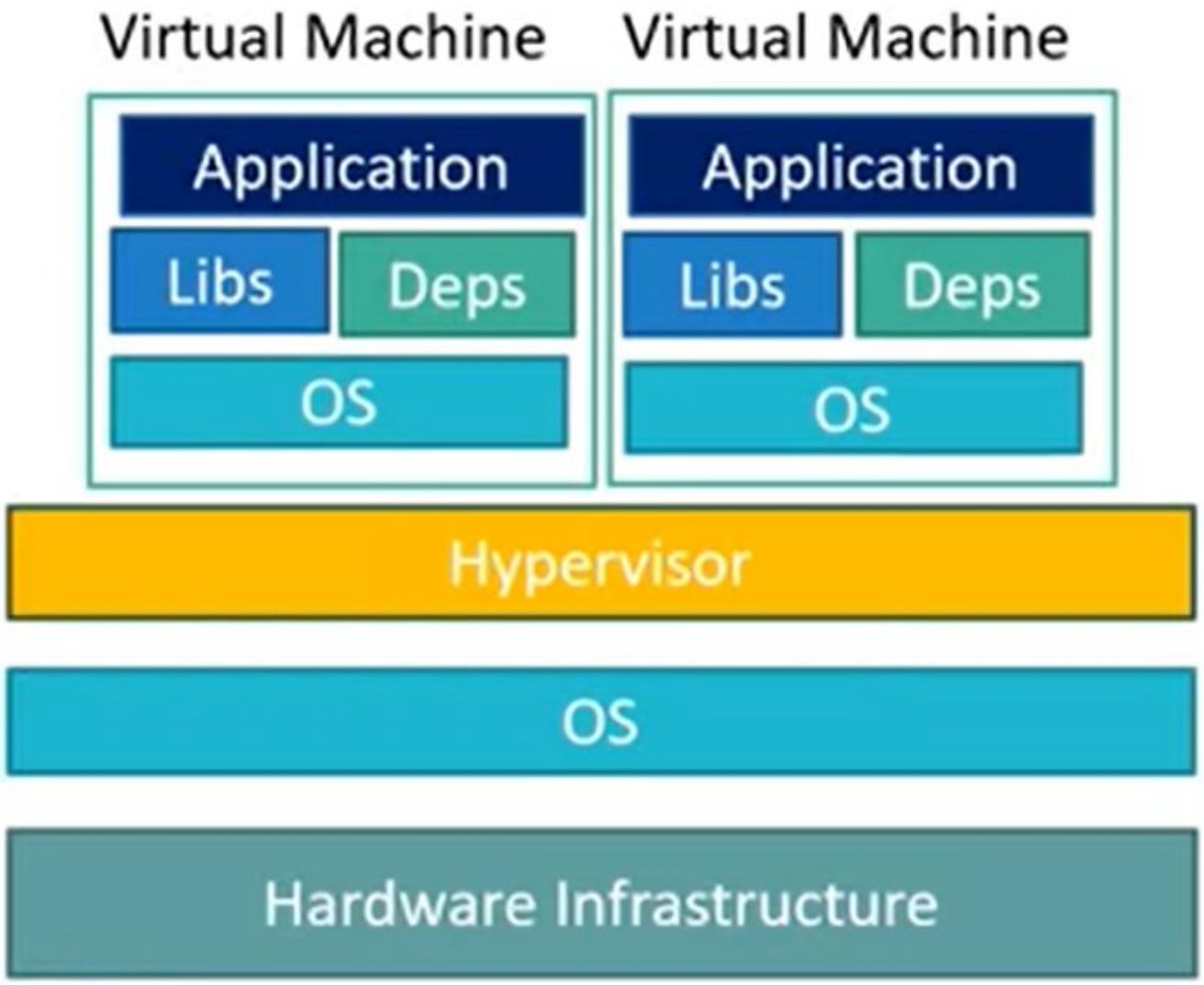
Needs for Containers

- To scale up your application you can't do it on the same VM because the port is alrady in use





VMs

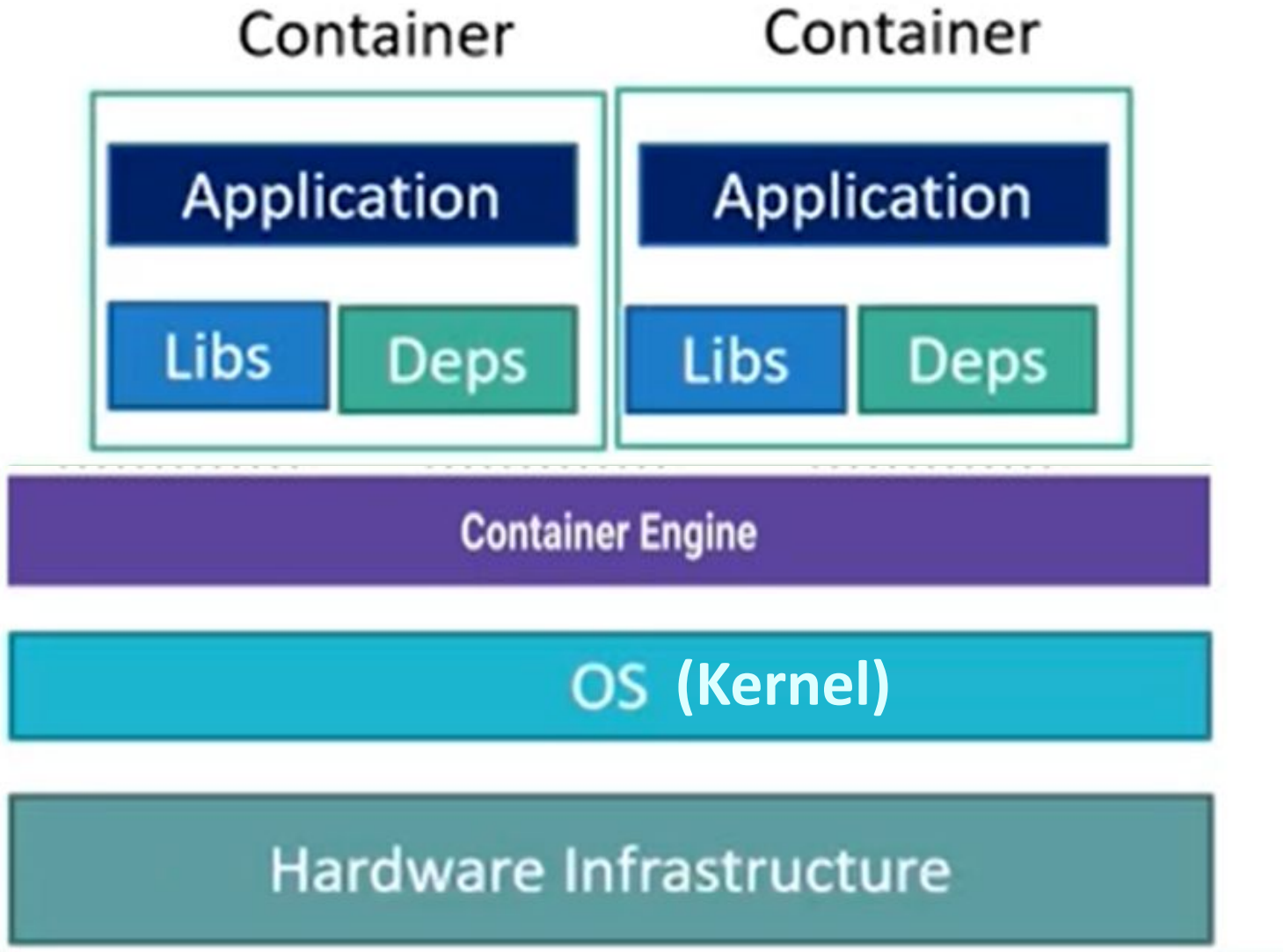

Virtual Machine
Application
Libs
Deps
OS
Virtual Machine
Application
Libs
Deps
OS
Hypervisor
OS
Hardware Infrastructure

Reflexion

- Whouldn't be nice if developer could package and deliver the application along with all the dependencies installed in a sandbox and run it Whatever enviroenemnt we want?
- What if i could create a sandbox environment inside my virtual machine that allows me to isolate applications from each other without having compatibility issues ?

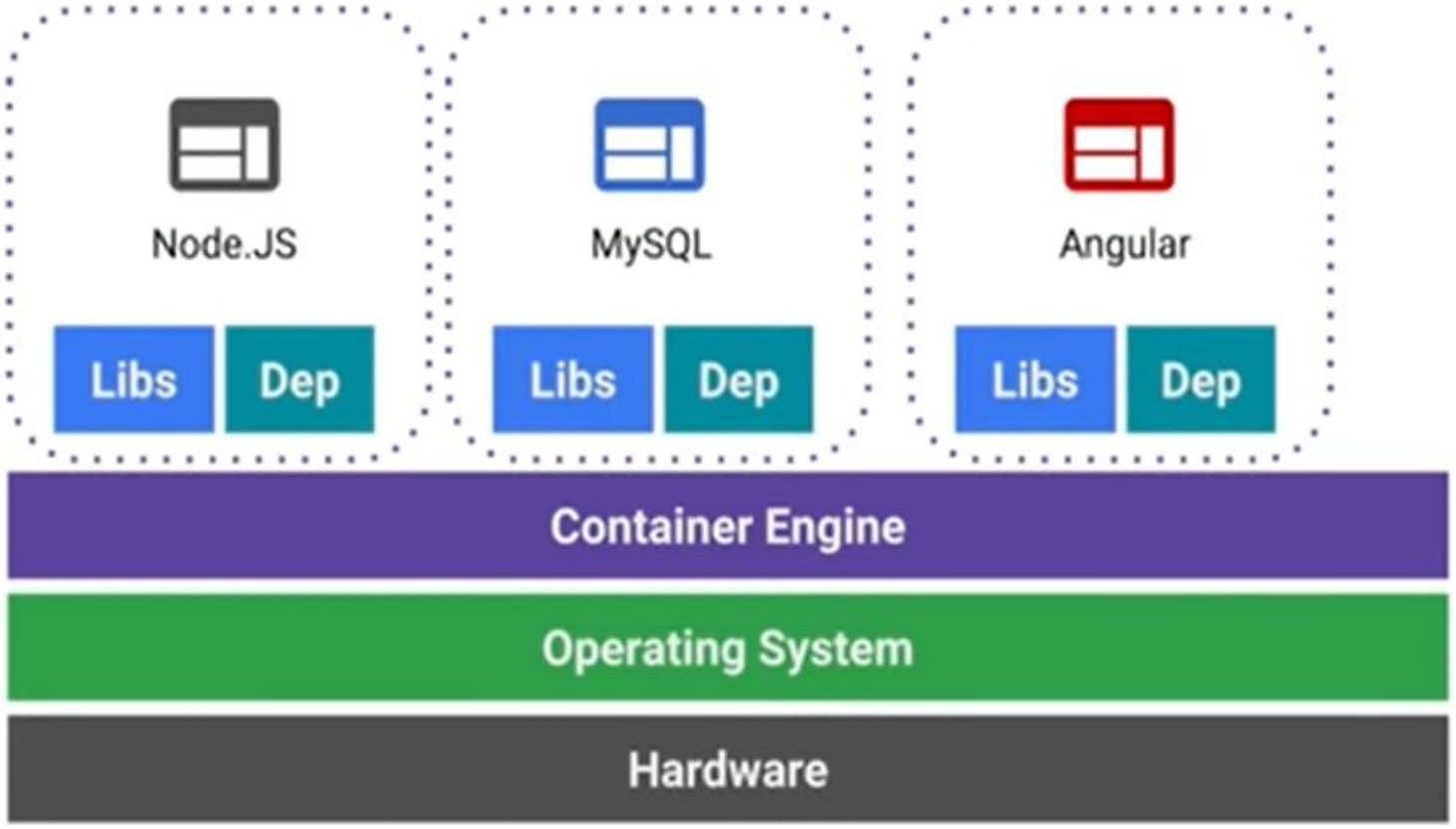
docker

Containers



docker

Containers vs. VMs

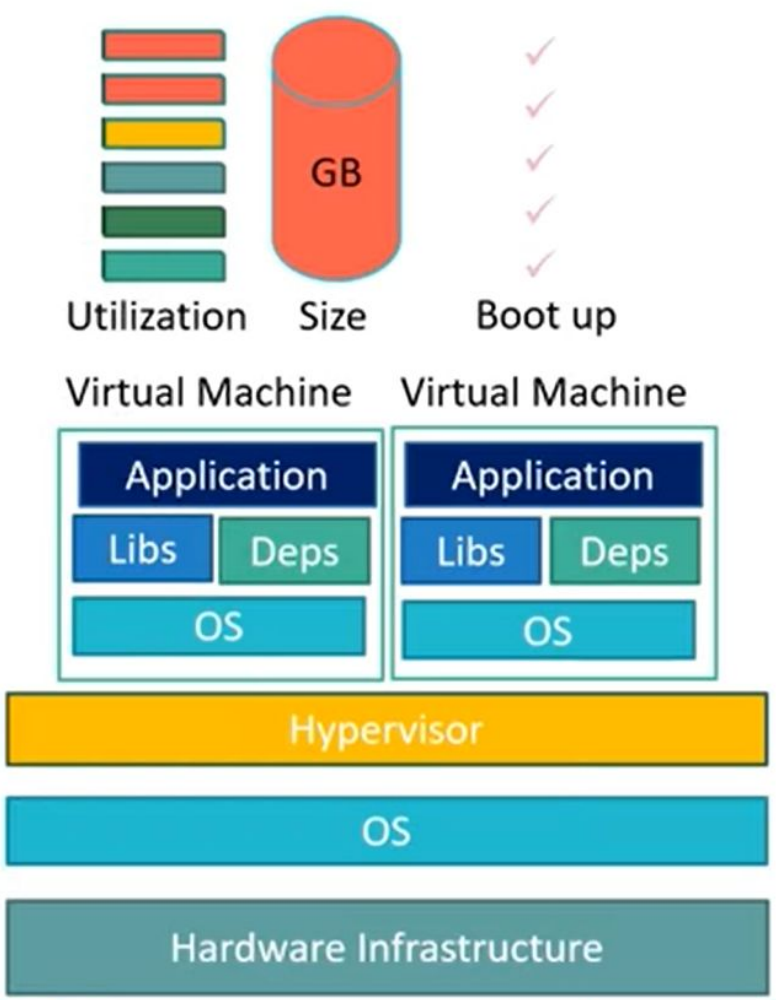


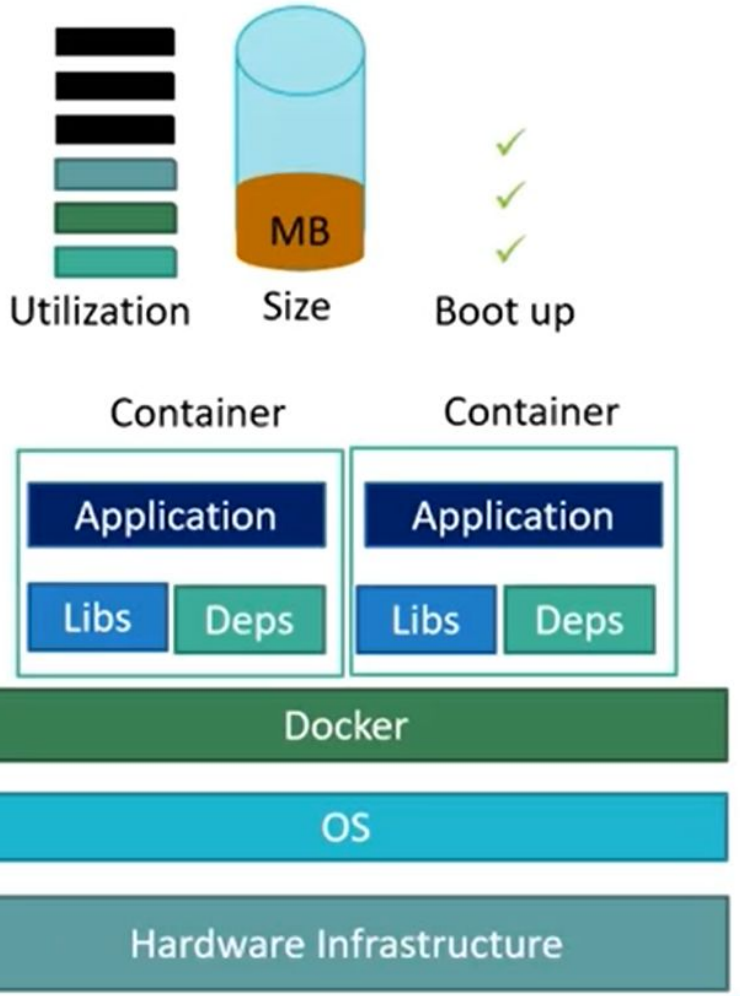
Containers = OS - kernel - unwanted Libs/Bins

Containers = App + Deps

docker

Containers vs. VMs

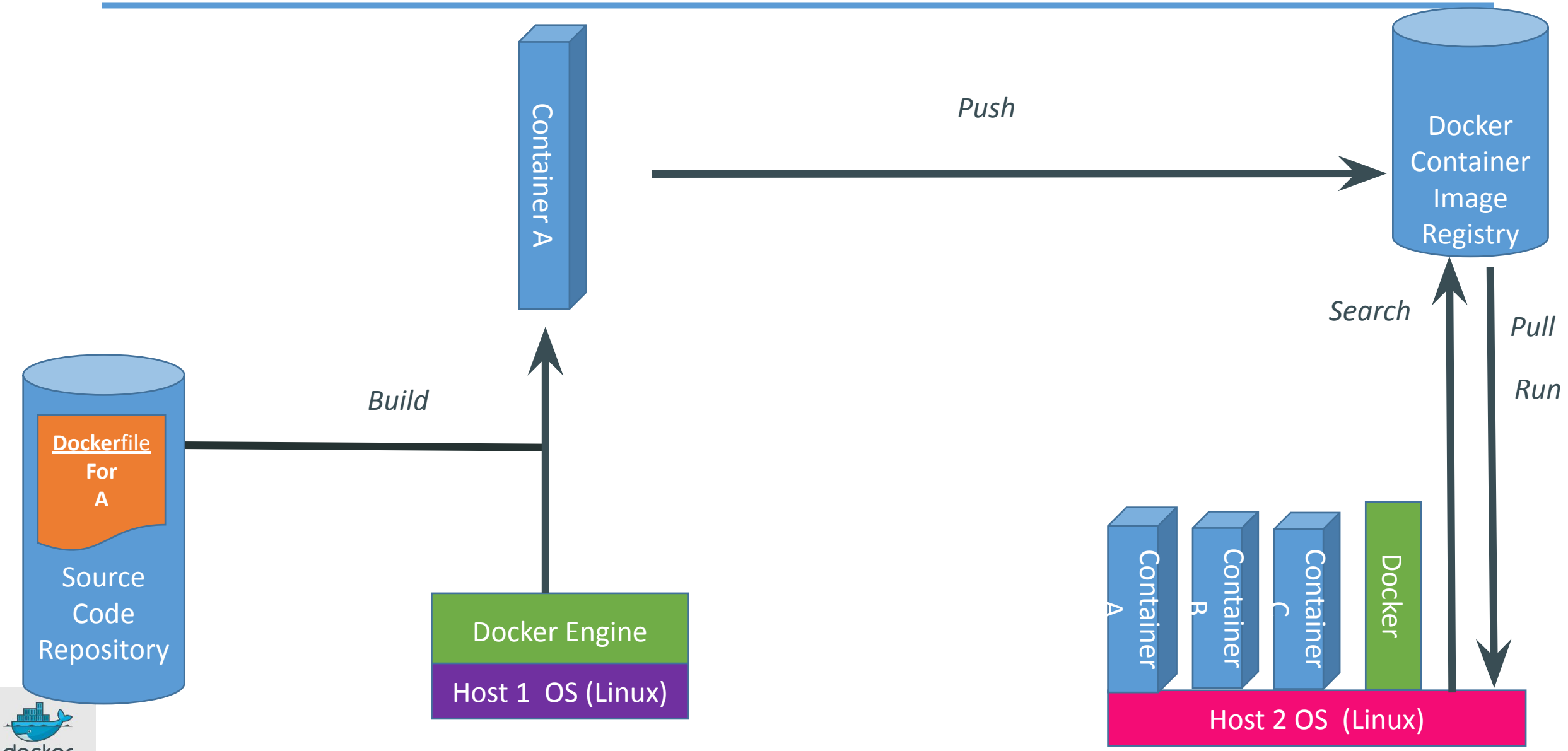




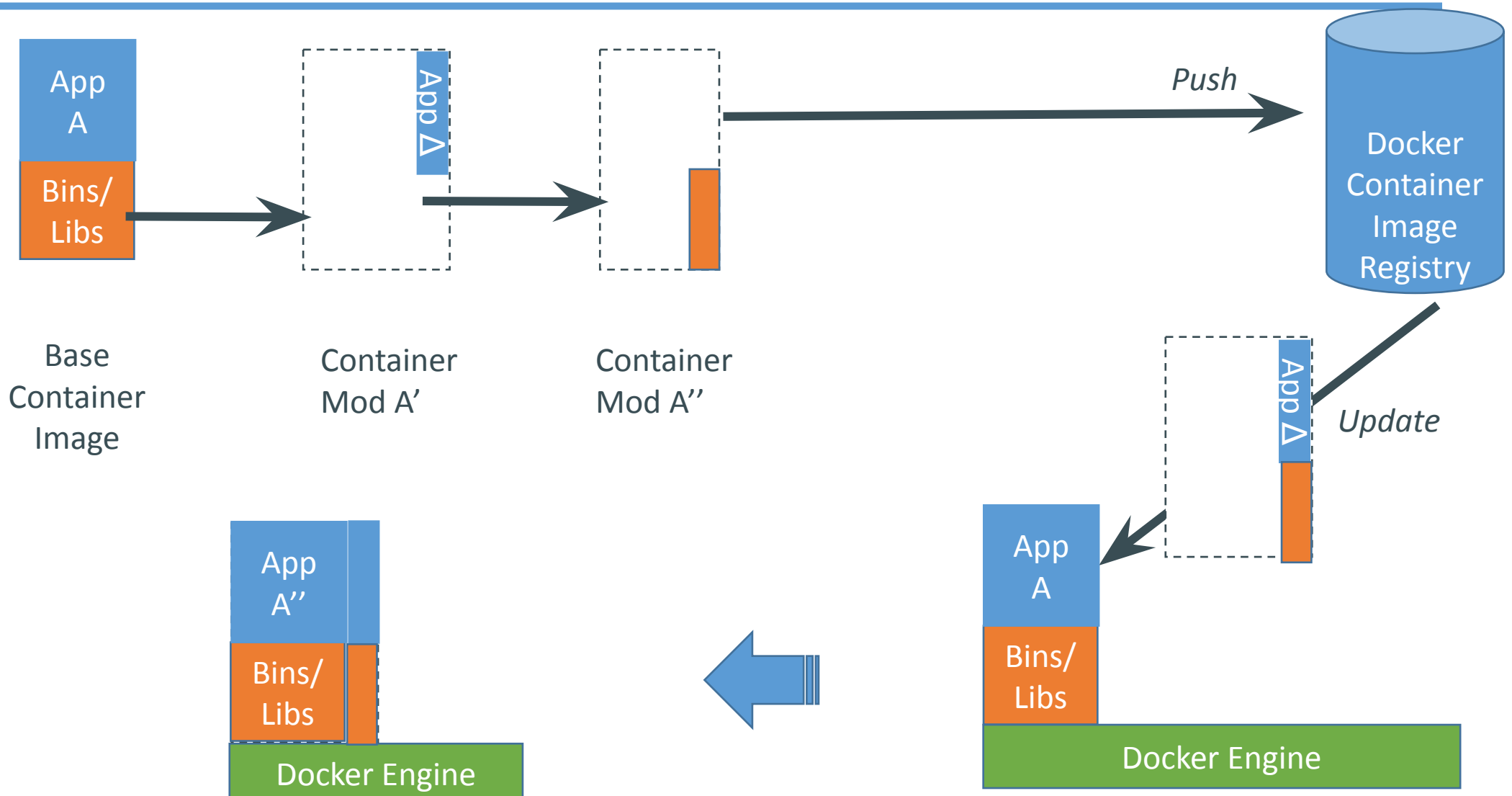
Run anywhere
Lighter
Less memory & CPU usage
Faster at start-up

docker

What are the basics of the Docker system?



Changes and Updates



Host is now running A''

Host running A wants to upgrade to A''.
Requests update. Gets only diffs

docker