Present by
Amit Ganvir

git

GitHub

RED HAT
CERTIFIED
ENGINEER

MICROSOFT
CERTIFIED
Systems Administrator

KUBERNETES
CERTIFIED
Administrator

AWS
CERTIFIED
Solutions Architect
Associate

AMIT GANVIR

Amit D. Ganvir, a DevOps Engineer in the IT sector since 2014, started writing this book in 2018. He hated writing a script but finally decided to learn when he saw scripting differently, and one quote changed everything "Scripting is nothing but the collection of commands" So if we know the Linux commands and how to execute them, we can also write a script. He has worked on Linux since 2008 and was a guest lecturer and technical trainer at Institute & College. This book will help us understand Linux and BASH scripting to automate the task, and it will fulfil the logic and which commands can help the automation task or program. He is an Engineer without engineering and lives in Nagpur, India

Technical Qualification

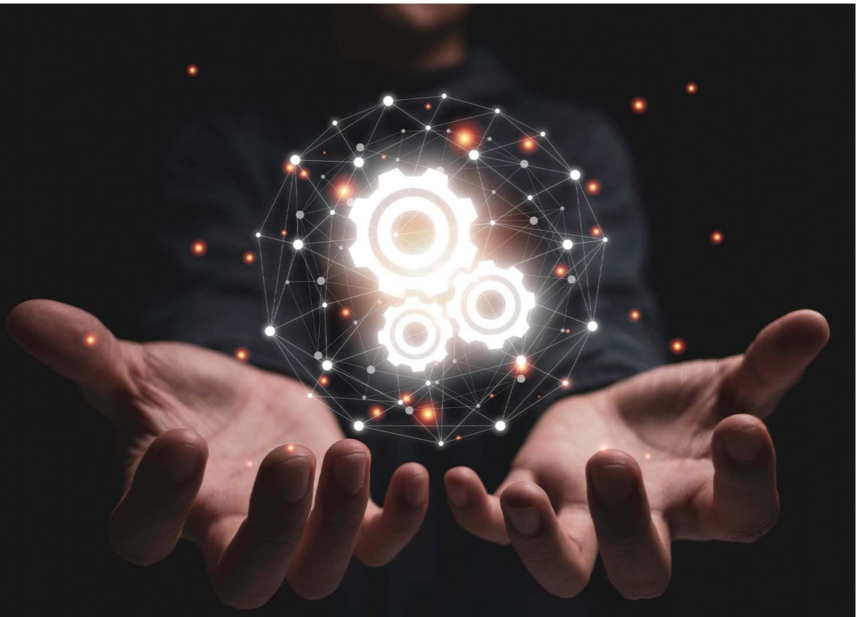
A+, N+, MCSA, CCNA, RHCT, RHCE, RHCVA, RHCSS, Kubernetes, Docker, Openshift, AWS, GCP, Terraform, Ansible, Helm, Jenkins, Gitlab and DevOps

BlueRose ONE.com
Stories Matter

INR 000

GENRE

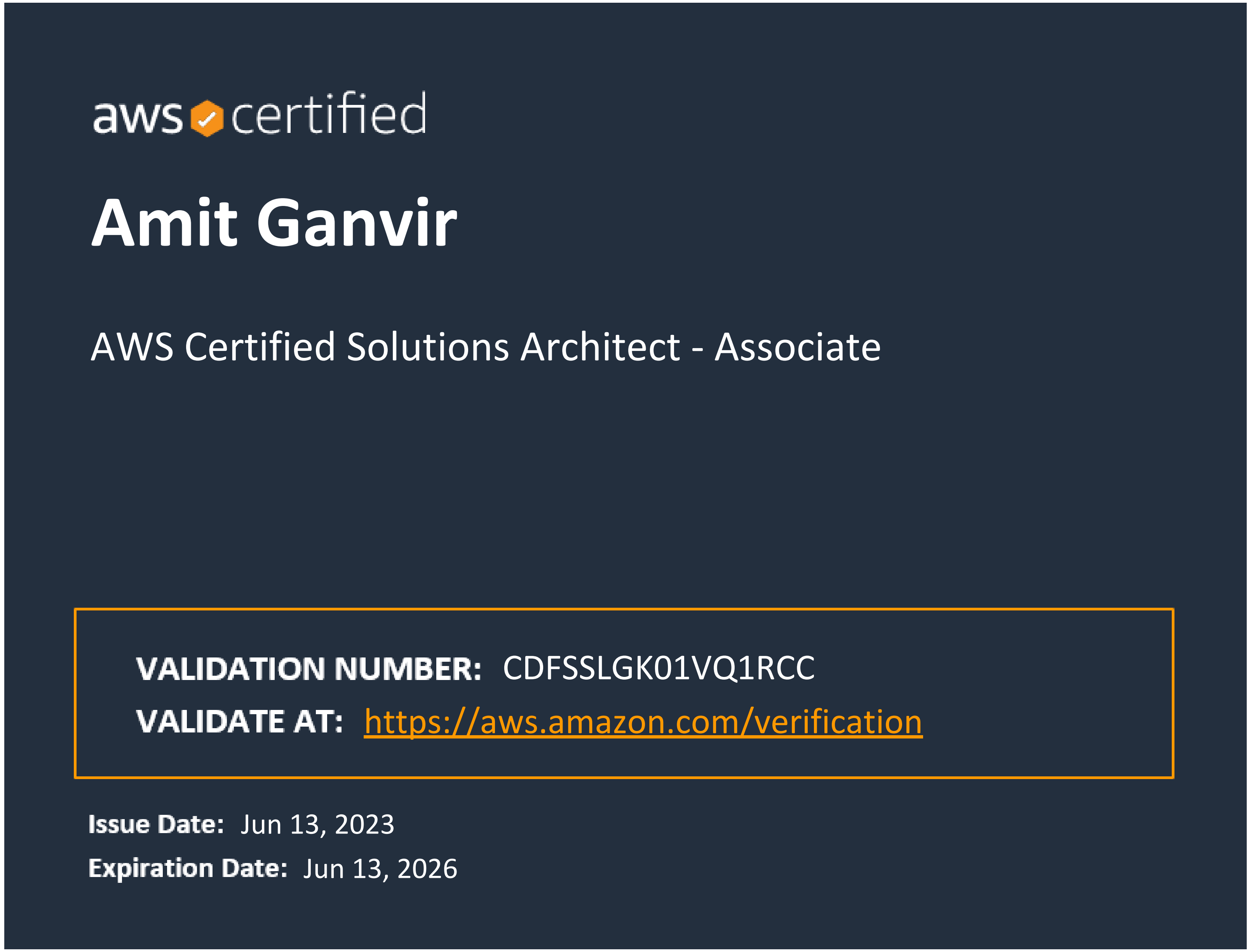
www.BlueRoseOne.com



ADVANCED SHELL SCRIPTING BOOK

Understand the power of GNU/Linux shell and become an expert

AMIT GANVIR



aws certified

Amit Ganvir

AWS Certified Solutions Architect - Associate

VALIDATION NUMBER: CDFSSLGK01VQ1RCC

VALIDATE AT: https://aws.amazon.com/verification

Issue Date: Jun 13, 2023

Expiration Date: Jun 13, 2026

CKA: Certified Kubernetes Administrator

ISSUED TO

AMIT GANVIR

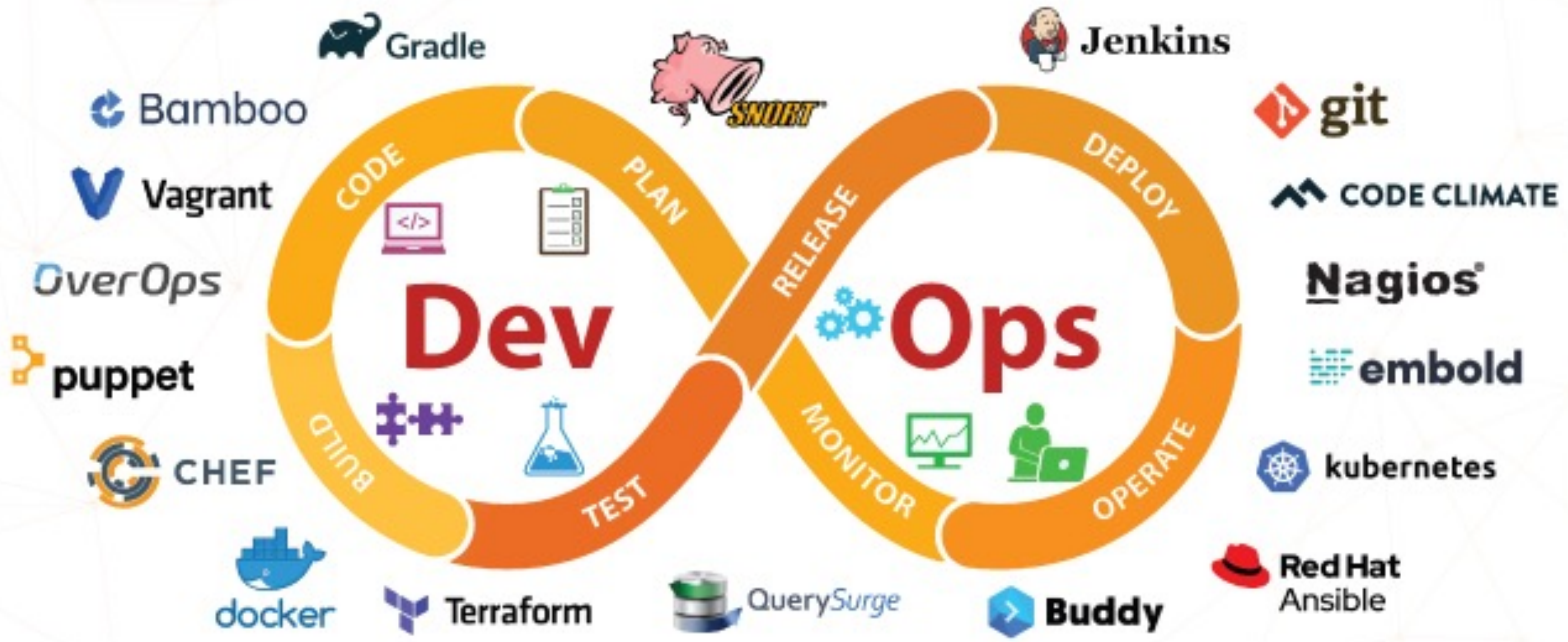
Issued on: 14 JUN 2023 | Expires on: 14 JUN 2026 | Issued by: The Linux Foundation
Verify: https://www.credly.com/go/0ymuTQQu

	Skill Set	Technologies
1	Cloud	AWS (Amazon Web Services) and GCP
2	Containerization Tools	Docker, Helm Chart, Kubernetes, Openshift & Min/(Kube/Shift).
3	Infrastructure As Code Software's	HashiCorp Terraform, Packer and Terragrunt
4	Operating System	RHEL, Ubuntu, CentOS and Windows Family
5	Configuration Management Tools	Ansible. Chef and Puppet.
6	Build Tools	NPM, Maven and Gradle
7	CI/CD Tools	Gitlab, Jenkins and pipeline groovy and DSL
8	Database	MYSQL, Mariadb, Mongodb, Couchbase and Cassandra Cluster Infra level
9	Scripting	BASH Shell Scripting and Python (Basic/Learning)
10	Version Controlling Tools	Gitlab, Bitbucket and Github
11	Application Server	Nginx, Apache and Tomcat
12	Monitoring	Zabbix, Kafka and zookeeper cluster Infra level
13	Service Discovery and Configuration	HashiCorp Consul and Git2consul
14	Virtualization Tools	Vagrant, Linux-KVM, VMware-Workstation and Virtual Box
15	Load Balancer	Nginx with Keepalived
16	Hypervisor	Qemu/KVM, VirtualBox, Vagrant
17	Software Storage	Ceph
18	Others	Nexus, Sonarqube, Jfrog, Jupeterhub, Rstudio, Keycloak and Vault Kubeflow, k8s Operator, Mlflow, Jfrog, Vault, Gitlab, Helm, Kubernetes, AWS, Cloudera Hadoop, Jupeterhub, Keycloak, Airflow, Rstudio, Argo, Kubeflow

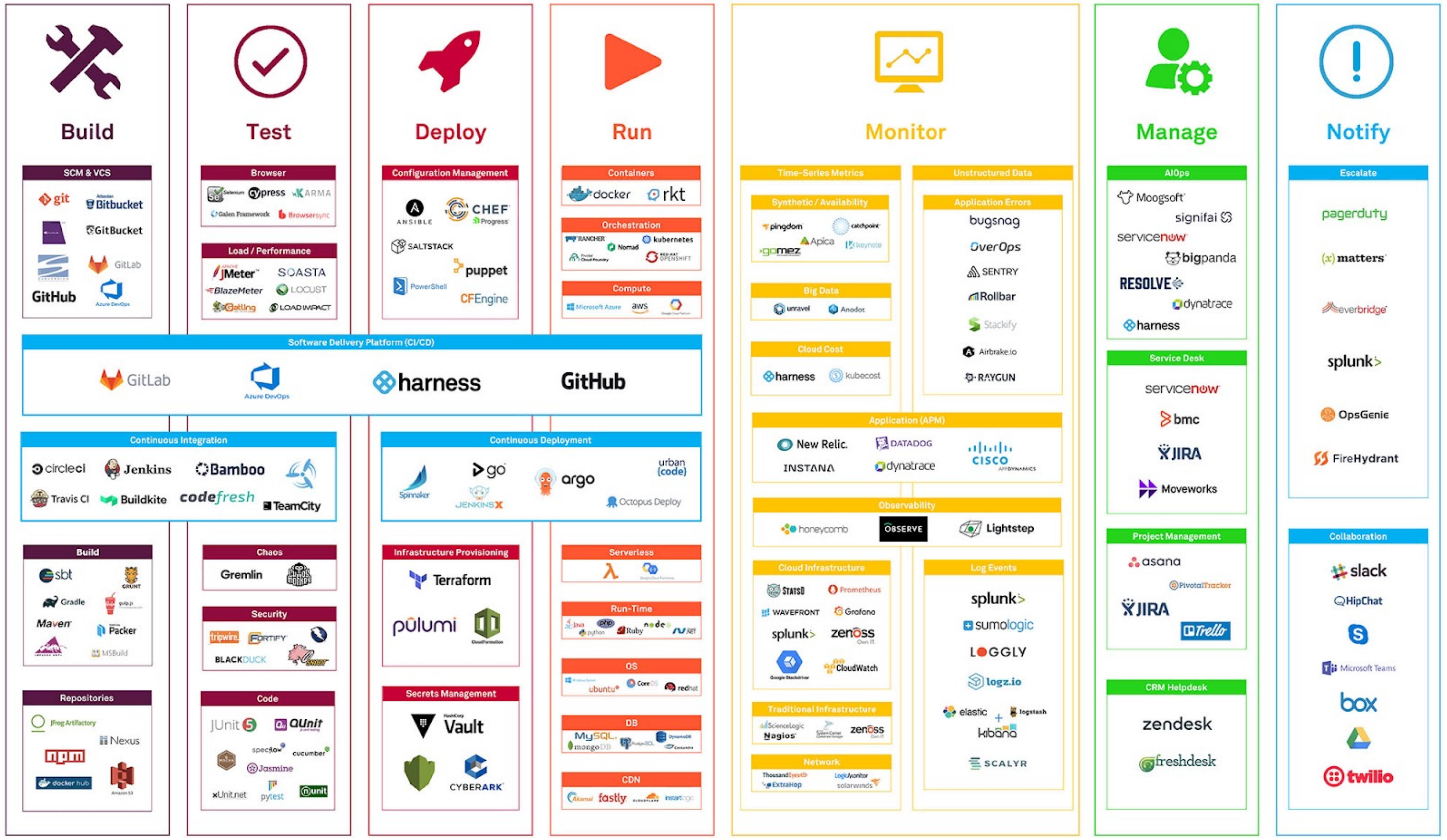
What is DevOps

- DevOps combines development and operations to increase the efficiency, speed, and security of software development
- It increases an organization’s ability to deliver applications and services at high velocity:
- evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes
- This speed enables organizations to better serve their customers and compete more effectively in the market

Best **DevOps Automation** Tools and Technologies



DevOps Tools Ecosystem 2021



GIT vs GitHub

git	GitHub
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

git
GitLab
GitHub
Bitbucket

What is version control?

- How version control helps high performing development and DevOps teams prosper

- Version control allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc

Benefits of version control?

- Having a GitHub repo makes it easy for you to keep track of collaborative and personal projects - all files necessary for certain analyses can be held together and people can add in their code, graphs, etc. as the projects develop

- Each file on GitHub has a history, making it easy to explore the changes that occurred to it at different time points

- You can review other people's code, add comments to certain lines or the overall document, and suggest changes.

How to Clone GitHub repo

Lets Clone our Repo and check the data

```
git clone https://github.com/amitganvir23/project-xyz.git
cd project-xyz
ls -l
cd ../
```

Lets Clone another Repo and check the data

```
git clone https://github.com/amitganvir23/kubernetes-minishift-openshift.git
ls -l
cd kubernetes-minishift-openshift
ls -l
```

GitHub

Step1:
Sign-up and Sign-In new account on Github https://github.com

Step2:
Create a new repository

git

Step1:
Install git for command line

Linux:

```
$sudo apt-get update
$sudo apt-get install git
```

Windows:

Install Gitbash
https://git-scm.com/downloads
OR
Install GitDesktop

Prerequisite before push the changes on repo

There are two ways

1) SSH – Using Public Key

Step1:

Create your public and private key using `ssh-keygen` command on your system

Step2:

Copy your public key

Step3:

Sign-In with your account on Github https://github.com

Step4:

Goto [Settings] –> [SSH and GPG Keys] –> [New SSH key] and Paste your public key

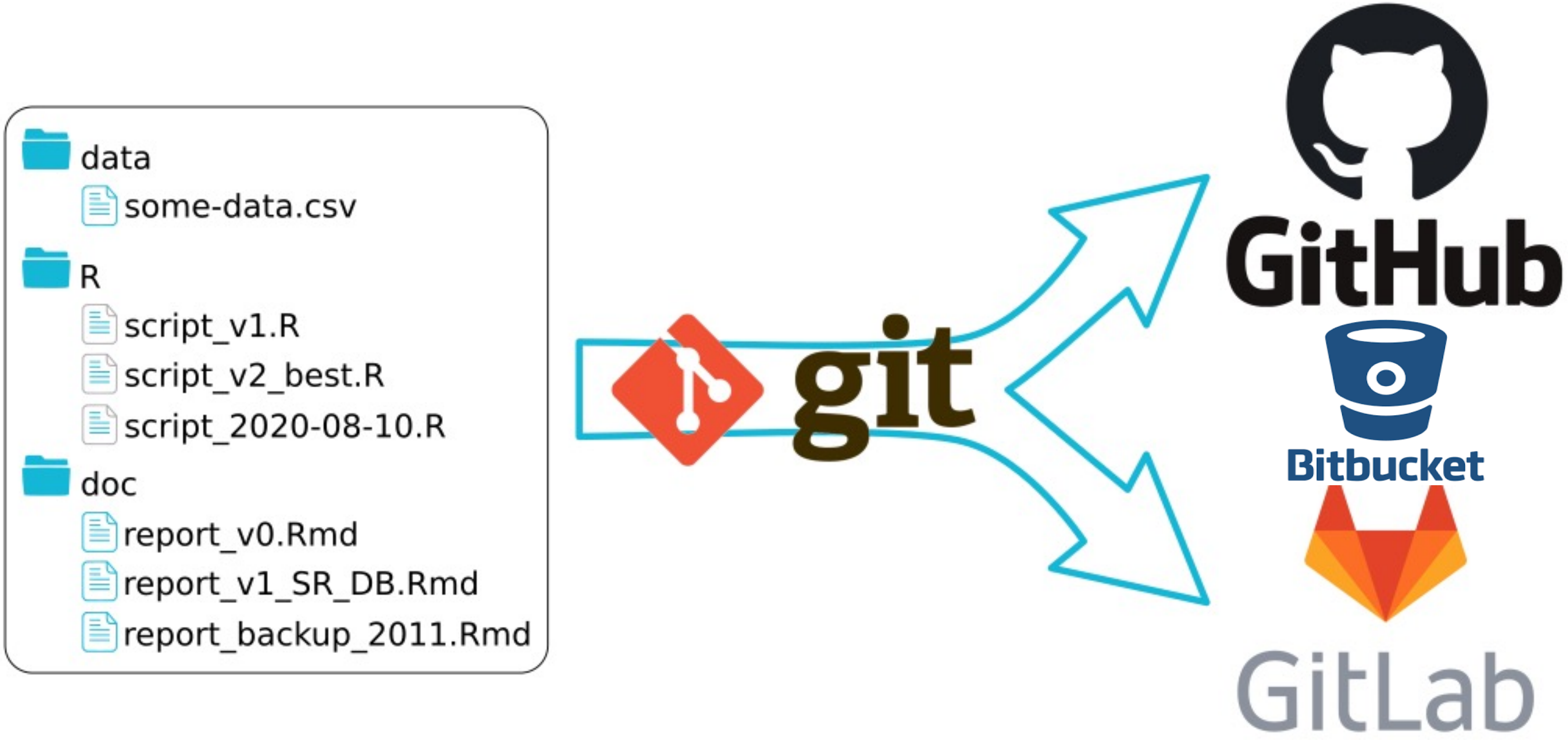
2) HTTPS – Using username and Token

Step1:

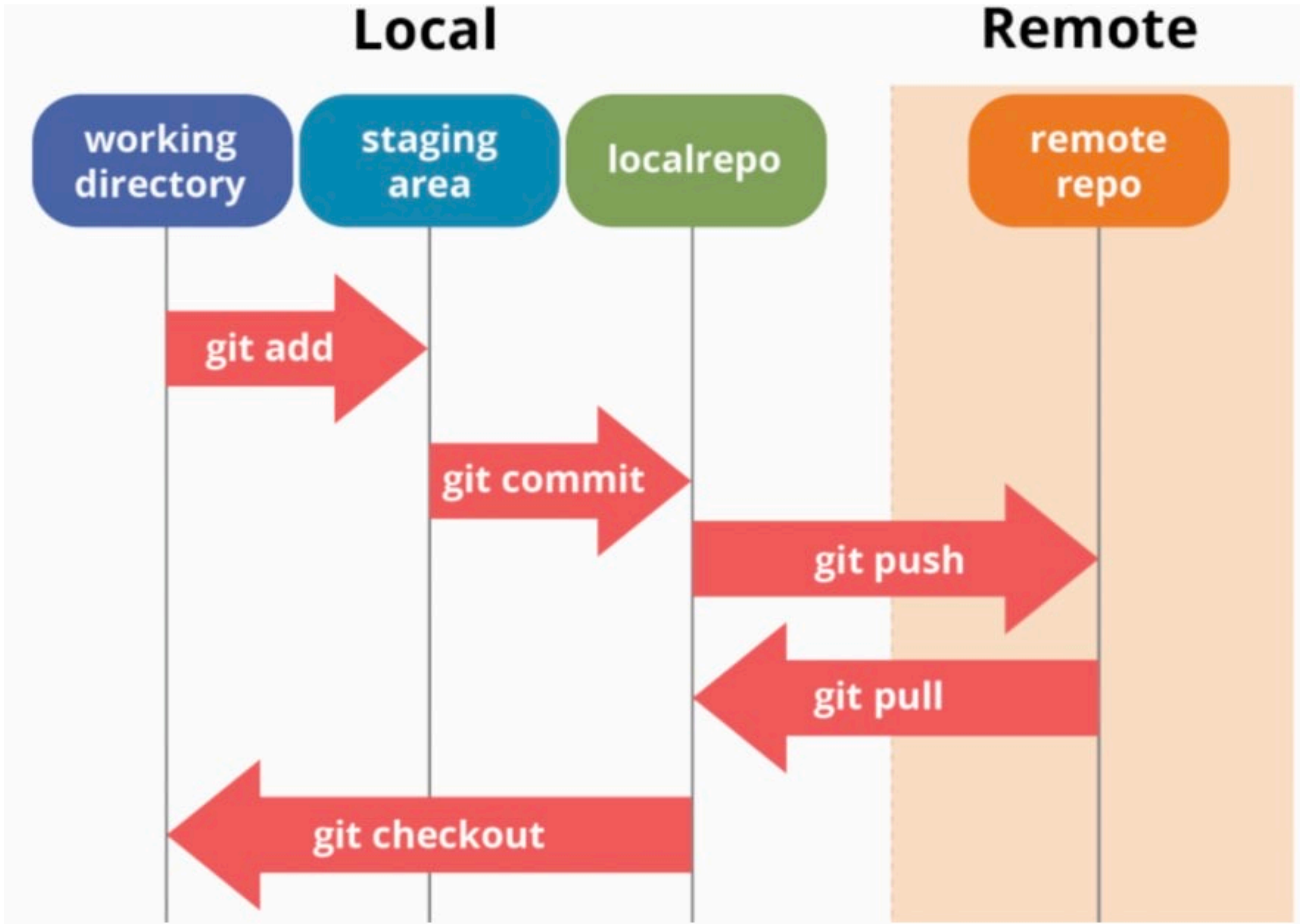
Goto [Settings] –> [Developer Settings] –> [Personal access tokens] –> [Tokens (classic)] –> [Generate new token (classic)]

Step2:

Copy your Token somewhere safe and it will use as a password for git push.


data
some-data.csv
R
script_v1.R
script_v2_best.R
script_2020-08-10.R
doc
report_v0.Rmd
report_v1_SR_DB.Rmd
report_backup_2011.Rmd
git
GitHub
Bitbucket
GitLab

How it Works?


Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout

What is Branch?

- In Git, a branch is a new/separate version of the main repository

Benefits of GIt

- With a new branch called new-design, edit the code directly without impacting the main branch
- EMERGENCY! There is an unrelated error somewhere else in the project that needs to be fixed ASAP!
- Create a new branch from the main project called small-error-fix
- Fix the unrelated error and merge the small-error-fix branch with the main branch
- You go back to the new-design branch, and finish the work there
- Merge the new-design branch with main (getting alerted to the small error fix that you were missing)

Prerequisite before push the changes on repo

SETUP

Step1:

Configuring user information used across all local repositories

```
git config --global user.name “[firstname lastname]”
```

Step2:

set a name that is identifiable for credit when review version history

```
git config --global user.email “[valid-email]”
```

How to store files on GitHub

Step1: Clone our Repo and goto the repo directory

```
git clone https://github.com/amitganvir23/project-xyz.git
cd project-xyz
```

Step2: Check your current branch name

```
git branch
```

Step3: Try to create a Sample file

```
echo amit > file1.txt
```

Step4: Check your file changes status

```
git status
```

Step5: Add your New/Untracked files and check your status

```
git add file1.txt           ## OR (git add .) (git add --all )
```

Step6: Clone our Repo and goto the repo directory

```
git commit -am “my first commit for testing”
```

Step7: Push all your current changes in your branch on remote repo

```
git push origin master
```

Additional Commands

`git diff` : Show difference between working directory and last commit.

`git log` : Display the entire commit history using the default format. For customization see additional options

`git reset --hard` : Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.

`git rebase -I <base>` : Interactively rebase current branch onto `<base>`. Launches editor to enter commands for how each commit will be transferred to the new base.

`git remote add <name> <url>` : Create a new connection to a remote repo. After adding a remote, you can use `<name>` as a shortcut for `<url>` in other commands.

`git pull <remote>` : Fetch the specified remote's copy of current branch and immediately merge it into the local copy

`git fetch <remote> <branch>` : Fetches a specific `<branch>`, from the repo. Leave off `<branch>` to fetch all remote refs

`git init <directory>` : Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository

`git checkout -b <branch>.` : Create and check out a new branch named `<branch>`. Drop the `-b` flag to checkout an existing branch.

Thank You!

https://ourcodingclub.github.io/tutorials/git/

In case of fire

1. git commit
2. git push
3. leave building