Git

Anil Joseph

Git

- Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Developed by Linus Torvalds in 2005.

Why Use Git

Version Control:	Tracks changes to files over time.
Collaboration:	Allows multiple people to work on the same project simultaneously.
Backup:	Provides a backup of your codebase.
Branching and Merging:	Supports complex workflows with branches and merge operations.

Presented by Anil Joseph

Key Concepts

 Repository (Repo):	A storage location for your project files and their history.
 Commit:	A snapshot of your project's state at a point in time.
 Branch:	An independent line of development.
 Merge:	Combining changes from different branches.
 Remote:	Repositories hosted on a server, e.g., GitHub, GitLab.

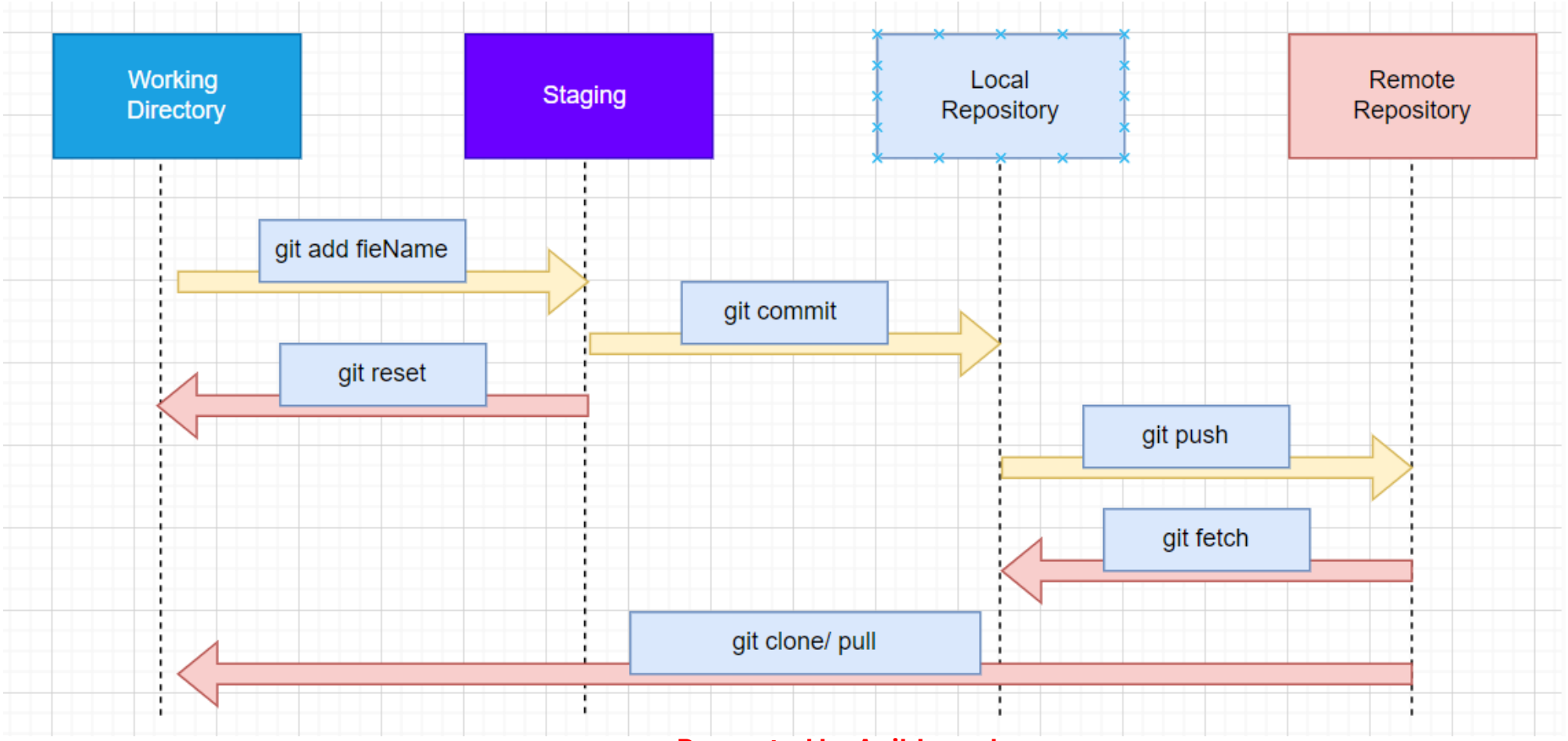
Presented by Anil Joseph

Basic Git Commands

- git init:
  - Initialize a new Git repository.
- git clone [url]:
  - Clone a repository into a new directory.
- git add [file]:
  - Add files to the staging area.
- git commit -m "message":
  - Commit changes with a message.
- git push:
  - Push changes to a remote repository.
- git pull:
  - Fetch and merge changes from a remote repository.

Presented by Anil Joseph

Git Flow


Working Directory
Staging
Local Repository
Remote Repository
git add fieName
git commit
git reset
git push
git fetch
git clone/ pull

Git Commands

 git status:	Displays the state of the working directory and the staging area.
 git restore [file]:	Discards changes in the working directory.
 git restore --staged [file]:	Unstages files.
 git reset [file]:	Unstages files.
 git stash:	Temporarily saves changes that are not ready to be committed.
 git stash pop:	Reapplies stashed changes.

Presented by Anil Joseph

Creating a Repository

Create a directory for your project.

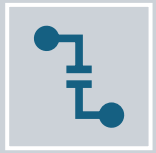
Initialize the repository with git init.

Add files and make your first commit.

Branching and Merging

Creating a Branch: git checkout -b [branch-name]

Switching Branches: git checkout [branch-name]



Merging Branches: git merge [branch-name]

Working with Remotes

Adding	Adding a Remote: git remote add origin [url]
Pushing	Pushing to a Remote: git push -u origin [branch-name]
Pulling	Pulling from a Remote: git pull origin [branch-name]

Jenkins

Presented by Anil Joseph

Jenkins

Jenkins is an open-source automation server written in Java.

Facilitates continuous integration and continuous delivery (CI/CD).

Supports the automation of building, testing, and deploying software.

Jenkins has a rich ecosystem of plugins to support building, deploying, and automating any project.

Jenkins can distribute build/test loads to multiple machines.

Web-based GUI, command-line interface, and easy-to-setup configuration.

Presented by Anil Joseph

History

Originally developed as Hudson in 2004.

Forked and renamed Jenkins in 2011 due to a dispute with Oracle.

Setup

- Prerequisites:
  - Java Development Kit (JDK): Ensure JDK is installed (Java 8 or later).
  - Operating System: Jenkins can be installed on various operating systems (Windows, macOS, Linux).
- Installation Steps:
  - Go to Jenkins Download Page.
  - Choose the appropriate package for your operating system.

Installation

- Windows:
  - Run the .msi installer.
  - Follow the setup wizard.
- macOS:
  - Use Homebrew: brew install jenkins-lts.
- Linux:
  - Debian/Ubuntu: sudo apt-get install jenkins.
  - Red Hat/CentOS: sudo yum install jenkins.

Initial Setup:

- Start Jenkins:
  - Windows: Jenkins runs as a service.
  - MacOS/Linux: Use command: sudo systemctl start jenkins or jenkins in terminal.
- Access Jenkins:
  - Open a web browser.
  - Navigate to http://localhost:8080.
- Unlock Jenkins:
  - Retrieve the initial admin password from the specified file (/var/lib/jenkins/secrets/initialAdminPassword).
  - Enter the password in the setup wizard.

Basic Configuration:

- Install Suggested Plugins:
  - Jenkins will recommend plugins based on common use cases.
- Create First Admin User:
  - Set up the admin username and password.
- Instance Configuration:
  - Configure the URL for Jenkins (if needed).

Plugins

- Jenkins plugins extend the functionality of Jenkins.
- Over 1,800 plugins available.
- Plugins can integrate with various tools, add new features, and enhance the automation process.
- Popular Jenkins Plugins:
  - Git
  - Pipeline
  - Docker
  - JUnit

Tools

- Jenkins integrates with various tools to streamline development workflows.
- Tools can be categorized into build tools, testing tools, deployment tools, and monitoring tools.
- Build Tools: Maven, Gradle, Ant
- Testing tools: Junit, Selenium
- Deployment Tools: Docker, Kubernetes
- Monitoring Tools: Prometheus, Grafana

Pipeline

- Jenkins Pipeline is a suite of plugins that support implementing and integrating continuous delivery pipelines.
- A pipeline defines the entire CI/CD process in a script.

Types of Pipelines

- Declarative Pipeline:
  - Uses a simplified, predefined syntax.
  - Easier to write and maintain.
- Scripted Pipeline:
  - Uses Groovy-based DSL (domain-specific language).
  - Offers more flexibility and control.

Key Pipeline Concepts

- Stages:
  - Represents a phase of the pipeline (e.g., Build, Test, Deploy).
- Steps:
  - Individual tasks within a stage (e.g., executing a shell command).
- Agents:
  - Specifies where the pipeline or stage runs (e.g., any, specific node).
- Post Actions:
  - Defines actions to take at the end of a pipeline or stage (e.g., cleanup, notifications).

Benefits of Using Jenkins Pipeline

- Code as Configuration:
  - Pipelines are defined in code, making them version-controlled and easy to manage.
- Complex Workflows:
  - Support for complex build, test, and deployment workflows.
- Extensibility:
  - Integrate with various tools and customize pipelines as needed.

Jenkins Nodes

- Nodes are machines that Jenkins uses to execute jobs.
- Consist of a master node (controller) and multiple agent nodes (build slaves).
- Master Node (Controller):
  - Central machine that manages Jenkins.
  - Handles job scheduling, monitoring, and dispatching builds to agent nodes.
- Agent Nodes (Build Slaves):
  - Machines that perform the actual build, test, and deployment tasks.
  - Can be physical machines, VMs, or containers.

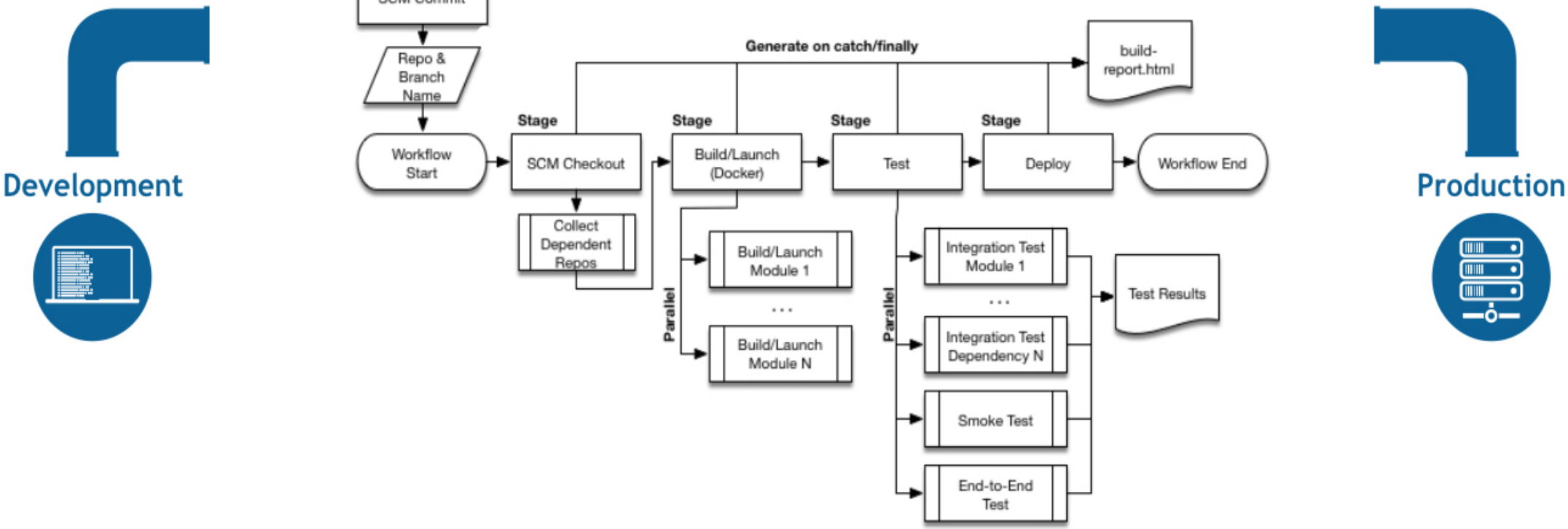
Jenkins Jobs

- Jenkins jobs are the fundamental units of work in Jenkins.
- Define tasks such as building, testing, and deploying code.
- Freestyle Project:
  - Basic job type.
  - Configured through the web interface.
  - Supports SCM, build triggers, build steps, and post-build actions.
- Pipeline
  - Defined using a Jenkinsfile.
  - Supports complex workflows and stages.

Jenkins Jobs

- Multi-branch Pipeline:
  - Manages multiple branches in a single repository.
  - Automatically creates pipelines for each branch.
- Folder:
  - Organizes jobs into folders for better management.
- External Job:
  - Tracks the execution of jobs run outside Jenkins.

CD Flow



Presented by Anil Joseph

Thank You

anil.jos@gmail.com

Presented by Anil Joseph