University of Colorado **Boulder**

Network Management and Automation

CI/CD & Unit Testing

Levi Perigo, Ph.D.
University of Colorado Boulder
Department of Computer Science
Network Engineering

Review

End world hunger
End world poverty
Automate Everything

Benefits of Automation

- **Versionable**
  - Change management
    - ***GitHub***
  - Records/audit
  - CI/CD

- **Repeatable & customizable**
  - Templates, Playbooks, Cookbooks, etc.

- **Testable**
  - Jenkins, Travis, Batfish, Pytest, Custom

- **Rapid deployment**

Benefits of Automation

- **Automated back-up system**
  - "Single" point of control = one point of back-up for configurations, templates, etc.
    - ***Logically centralized, physically distributed***
  - Additional layer of backup
    - ***Hosted in VM***
      - Backup VM = backup ENTIRE network!

- **Forget about manually configuring devices**
  - ("conf t" is dead)

Goal of Automation

- **"Single" point of control (NMAS)**
  - Drive the network from management station
    - ***Configuration change (VLAN, Route)***

- **Automation tools**

- **Run infrastructure as code (IaC)**
  - What does this mean?

Automation Framework

- **Scalable**
  - Proof of Concept (PoC)
    - ***Solve for few, can solve for many***
      - *Code must change at large scale

- **Easily configurable and customizable**

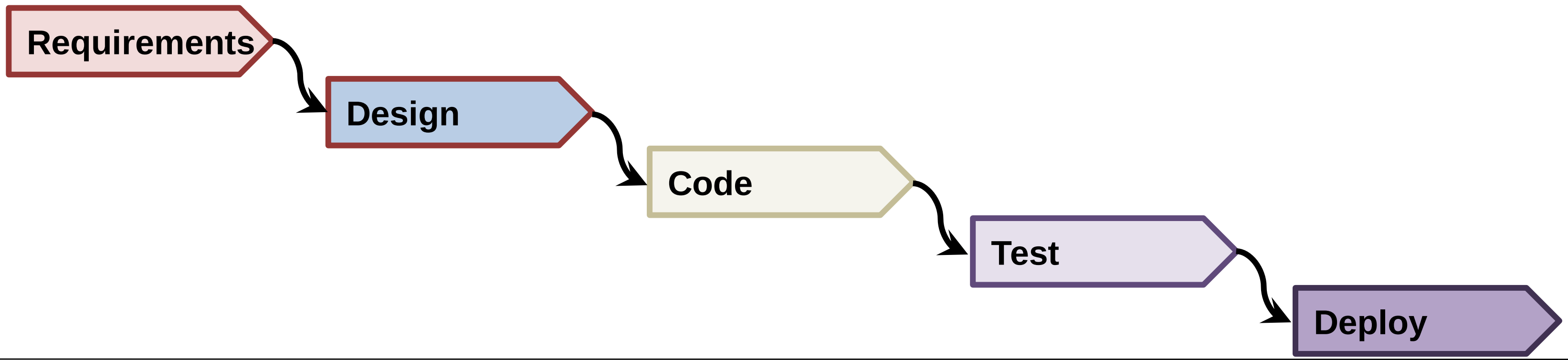
- **Config. verification and enforcement**

- **Statistics collection**

How do we use automation framework for CI/CD and testing?

Traditional Software Development Model

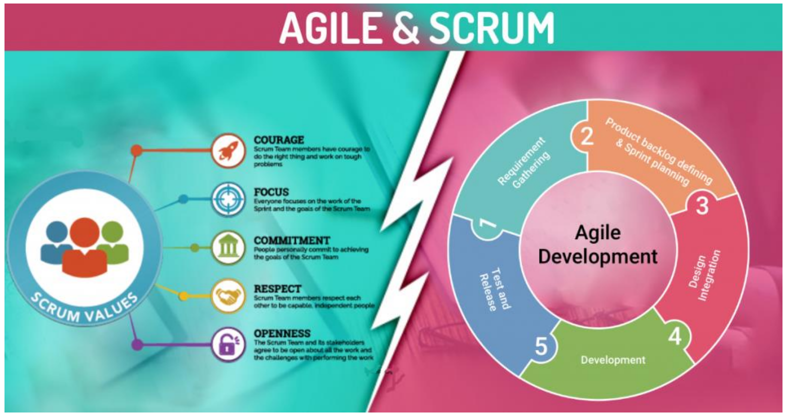
- **The waterfall software development model**
  - Each development phase begins only after the previous phase is complete
  - Codebase issues may not be discovered until deployment
  - Considered inflexible and slow



Agile Software Development Model

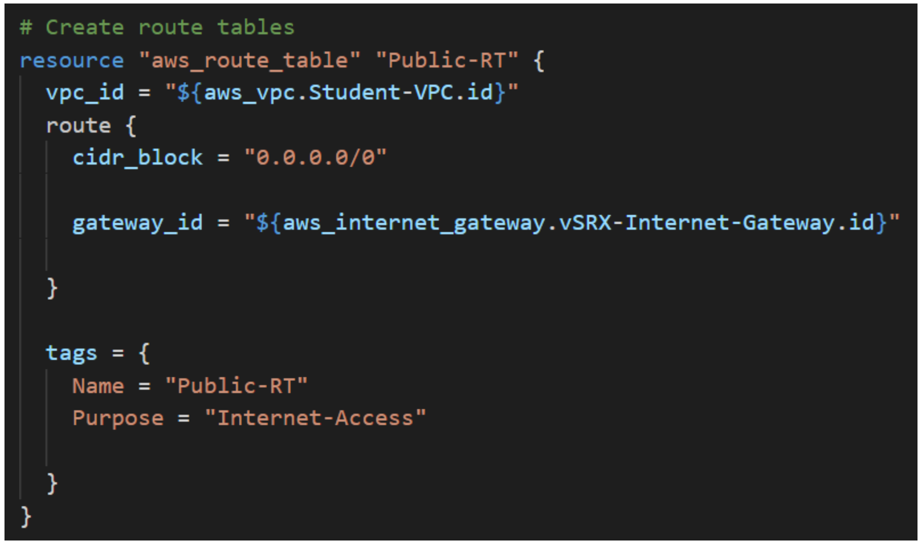
- **Continuous, reliable software delivery**
  - Provides high customer satisfaction
    - ***Frequent releases***
    - ***Welcomes change***
    - ***Frequent communication between developers and stakeholders***
    - ***Teams reflect on ways to become more effective and adjust accordingly***
    - ***Manifesto for Agile software development http://agilemanifesto.org***

  - Values
    - ***Individuals and Interactions over processes and tools***
    - ***Working software and comprehensive documentation***
    - ***Responding to change over rigid plans***

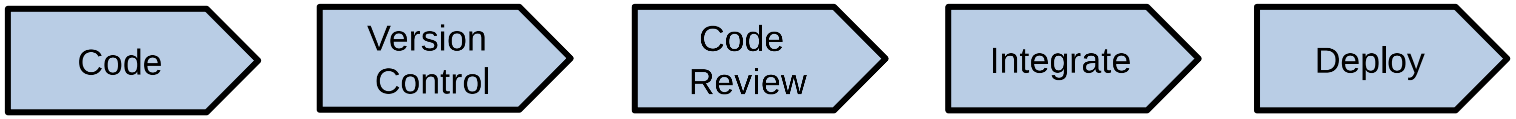

AGILE & SCRUM
SCRUM VALUES
COURAGE
Scrum Team members have courage to do the right thing and work on tough problems
FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team
COMMITMENT
People personally commit to achieving the goals of the Scrum Team
RESPECT
Scrum Team members respect each other to be capable, independent people
OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work
Agile Development
1 Requirement Gathering
2 Product backlog defining & Sprint planning
3 Design Integration
4 Development
5 Test and Release

Infrastructure as Code

- **DevOps principles and practices applied to infrastructure**
  - Operations provisions and manages network infrastructure using code
    - ***Enables rapid, consistent, and scalable deployment of infrastructure***
    - ***Enables development, test, staging, and production environments to be identically configured***

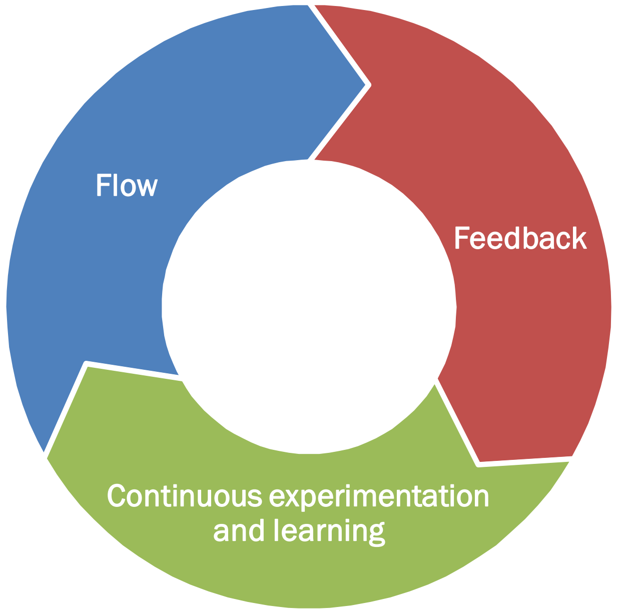


```
# Create route tables
resource "aws_route_table" "Public-RT" {
  vpc_id = "${aws_vpc.Student-VPC.id}"
  route {
    cidr_block = "0.0.0.0/0"

    gateway_id = "${aws_internet_gateway.vSRX-Internet-Gateway.id}"

  }

  tags = {
    Name = "Public-RT"
    Purpose = "Internet-Access"

  }
}
```



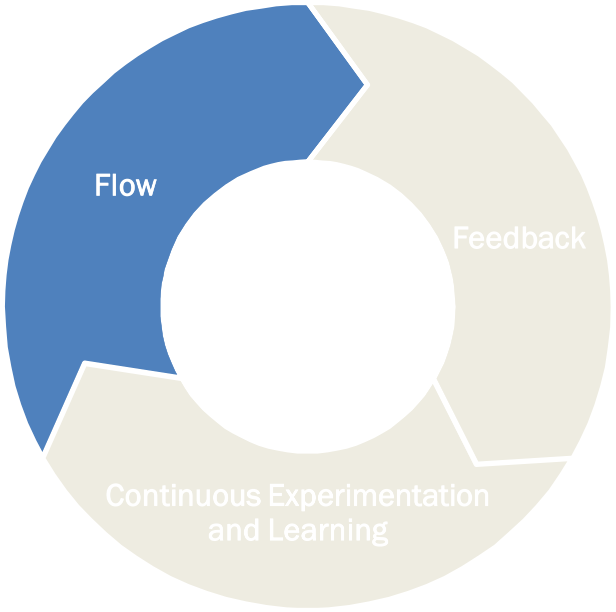
DevOps Culture

- **DevOps is a culture**
  - A set of shared principles, practices, and values
    - ***Flow, feedback, continuous experimentation, and learning***
    - ***People and communication are the keys***



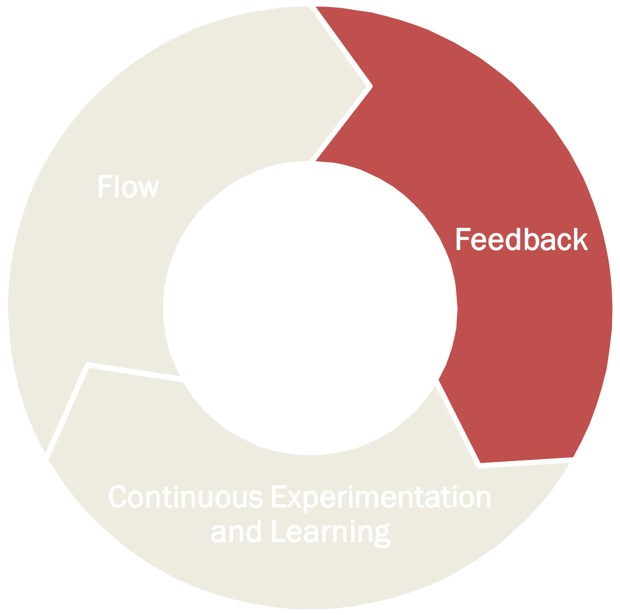
Flow

- **Consider the entire process**
  - From initial request to customer delivery
  - Do not permit local improvements to degrade the process as a whole
  - Always seek to increase rate of flow
  - Avoid technical debt



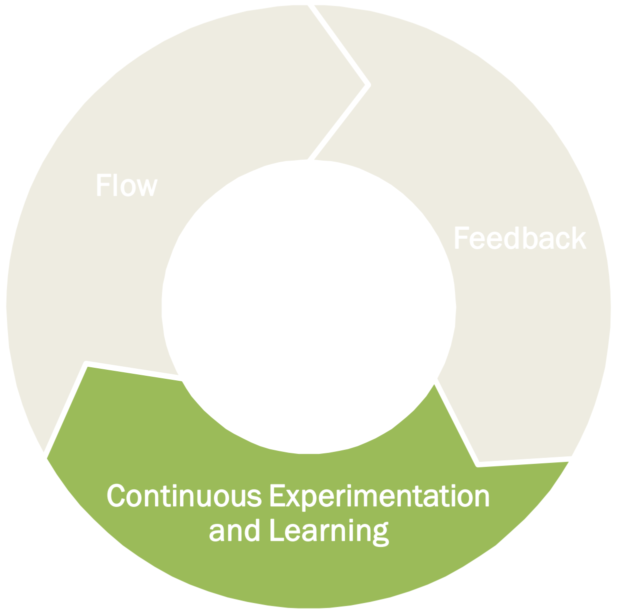
Feedback

- **Require, encourage, and accept feedback**
  - Use feedback to improve process flow, speed up delivery times, and reduce errors
  - Automate as much feedback as possible
  - Analyze feedback from each stage
    - ***Development***
    - ***Deployment***
    - ***Delivery***



Continuous Experimentation and Learning

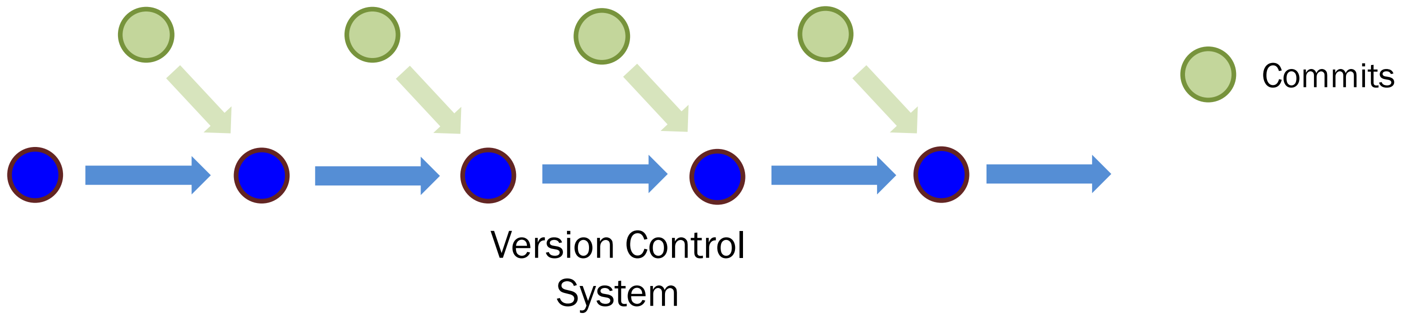
- **Continuous focus on improvement**
  - Allocate time for process improvement
  - Find fault with processes not people
  - Reward risk-taking
  - Introduce faults to stress-test the system



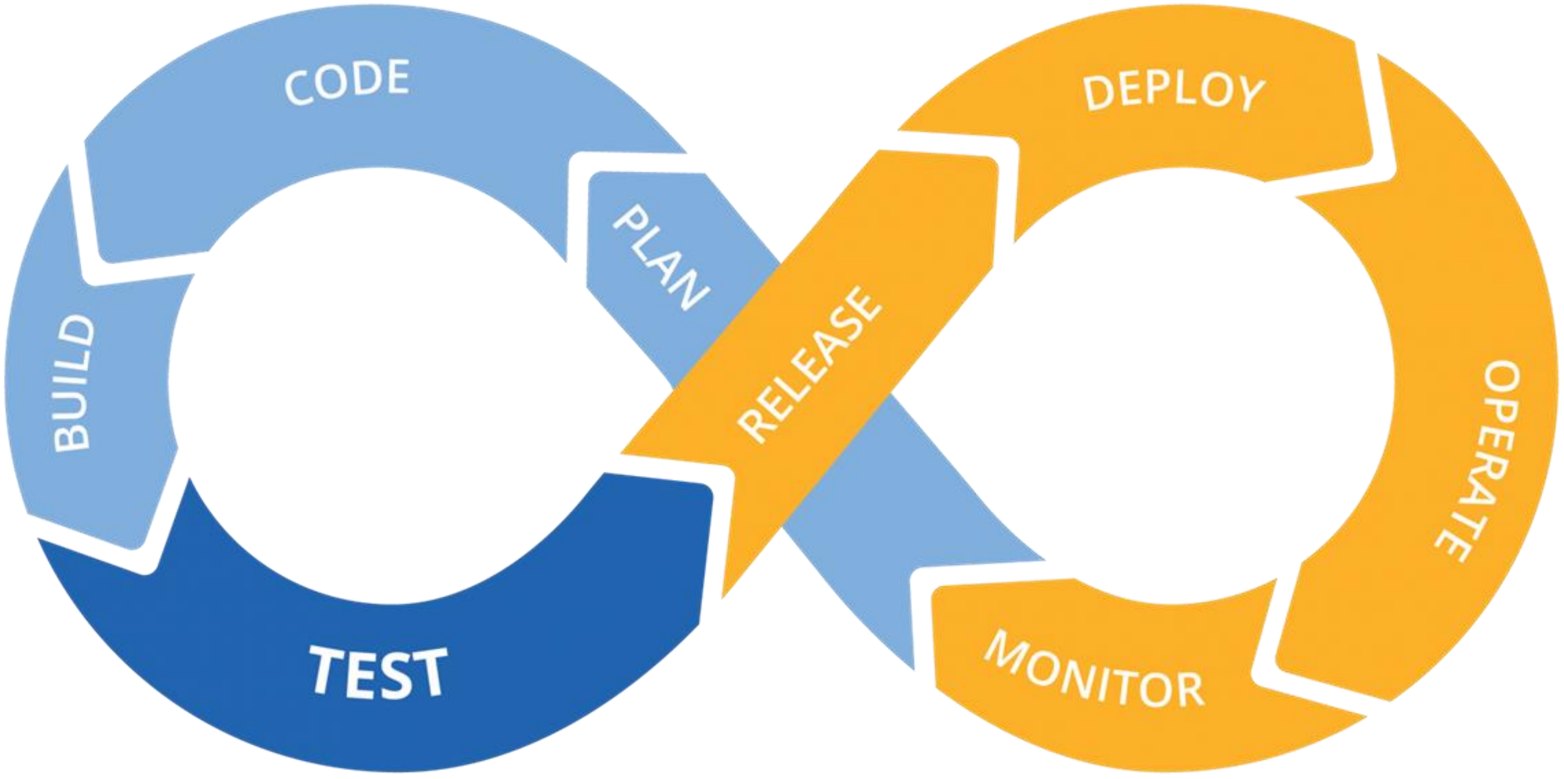
Continuous Practices

- **Continuous Integration**
  - Practice of frequently committing and <u>automatically</u> testing the code
  - Primary code branch remains production ready

- **Continuous Integration and Continuous Deployment**
  - Deliver and deploy frequently
  - <u>Development, test, stage, and production environments are as identical as possible</u>



CI/CD = Continuous Integration | Continuous Deployment and Delivery



What is Continuous Integration?

- **Continuous Integration** is a software development method where team members integrate their work at least once a day.
  - In this method, every integration is checked by an automated build to detect errors.

- This concept was first introduced over two decades ago to avoid “integration hell,” which happens when integration is put off till the end of a project.

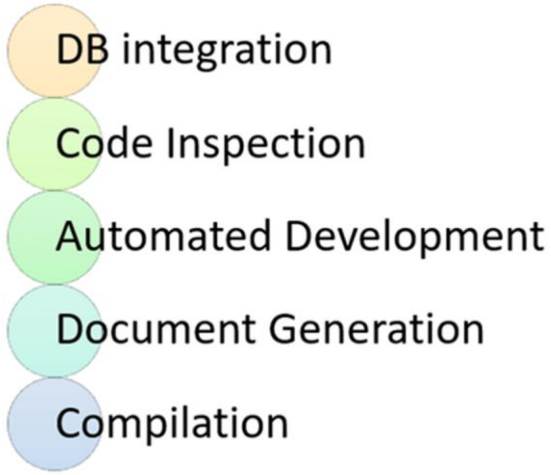
What is Continuous Integration?

- In CI after a code commit, the software is built and tested immediately.
- In a large project with many developers, commits are made many times during a day.
- Each commit code is built and tested. (How do we do this?)
  - If the test is passed, build is tested for deployment (where).
    - If the deployment is a success, the code is pushed to Production.
- This commit, build, test, and deploy is a continuous process, and hence the name continuous integration/deployment.

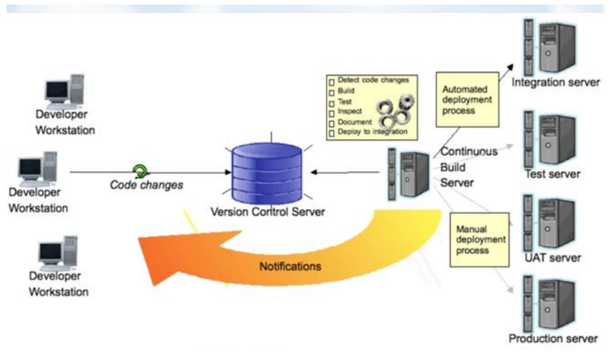
Development w/o CI	**Development with CI**
Lots of Bugs	Fewer bugs
Infrequent commits	Regular commits
Infrequent and slow releases	Regular working releases
Difficult integration	Easy and Effective Integration
Testing happens late	Continuous Integration testing happens early and often.
Issue raised are harder to fix	Find and fix problems faster and more efficiently.
Poor project visibility	Better project visibility

Compilation vs Continuous Integration

- **Compilation:**
  - Compilation is the process the computer takes to convert a high-level programming language code into a machine language that the computer can understand. It ensures a code compiler on every target platform.
    - ***Only compiles the code***
- **DB integration:**
  - Ensure DB and code in sync
  - Automated creation of DB and test data
- **Code Inspection:**
  - Ensures a healthy codebase
  - Identifies problems early and applies best practices
- **Automated Deployment:**
  - Allows you to release product anytime
  - Continually demo-able state and it works on any machine
- **Document generation:**
  - Ensure documentation is current
  - Removes burden from the developer
  - Produces build reports and metrics

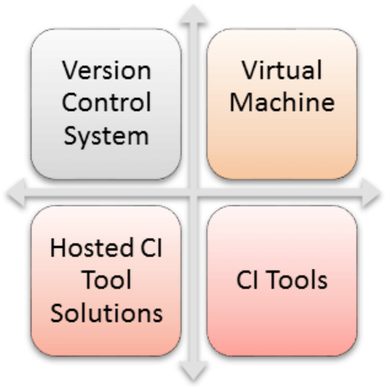


- **When do I build?**
  - Every check-in
  - Every time a dependency changes



- **How do I build?**
  - Ideally, the build should come from the command line and should not depend on IDE
  - The build should happen continuously using a dedicated CI server, not a cron job
  - CI build should be triggered on every check-in and not just at midnight
  - The build should provide immediate feedback and require no developer effort
  - Identify key metrics and track them visually
    - ***More importantly, act on them immediately***

What do I need to run CI?



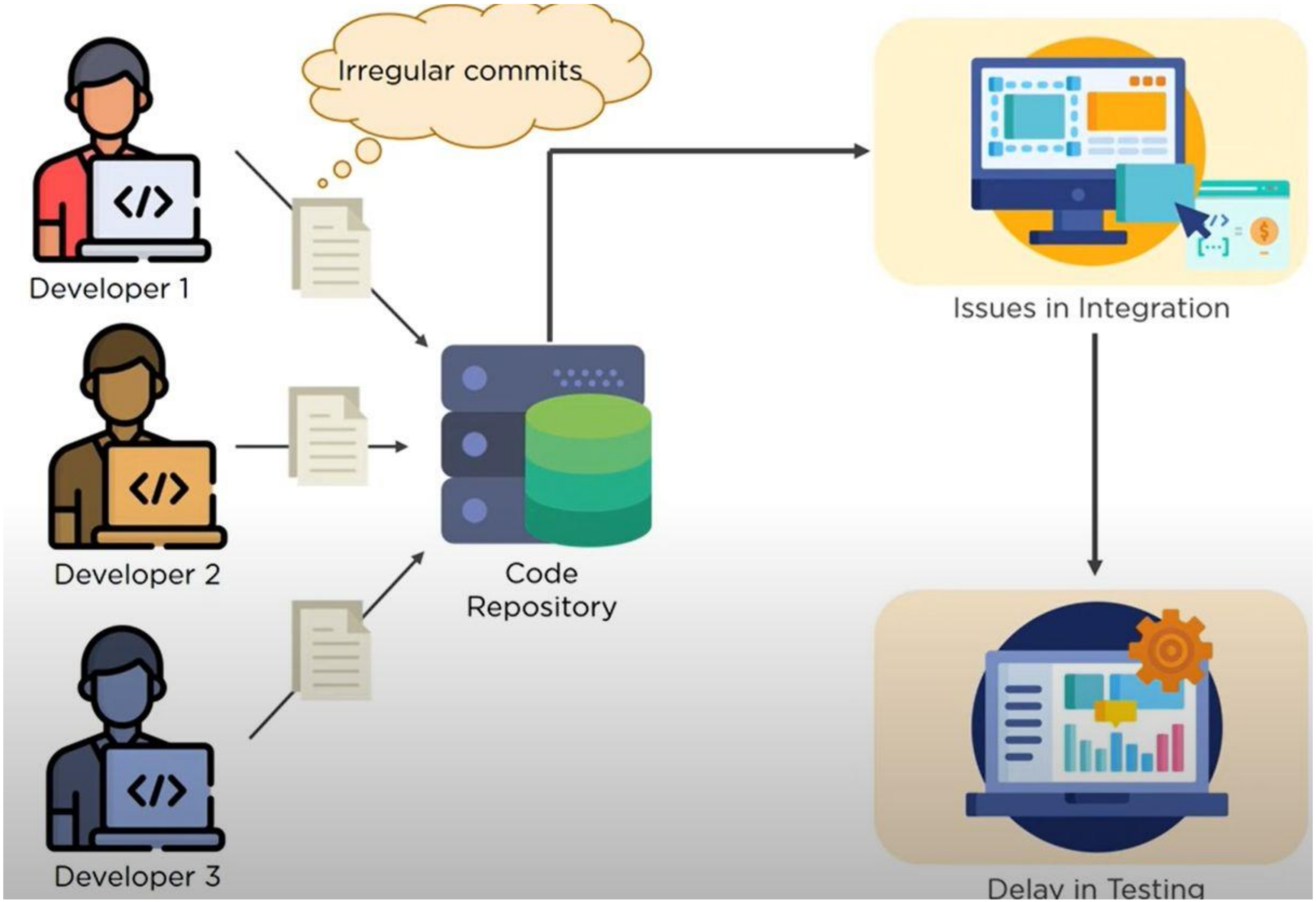
- **Version Control System (VCS):**
  - It offers a reliable method to centralize and preserve changes made to your project over time.
- **Virtual Machine:**
  - You should have a spare server or at least one virtual machine to build your system.
    - ***Logically centralized, physically distributed (hosted)***
- **Hosted CI Tool Solutions:**
  - To avoid servers or virtual machines, you should go for hosted CI tool solutions. This tool helps in the maintenance of the whole process and offers easier scalability.
- **Tools:**
  - If you select a self-hosted variant, you will need to install one of the many CI tools like: Jenkins, TeamCity, Bamboo, GitLab, etc.

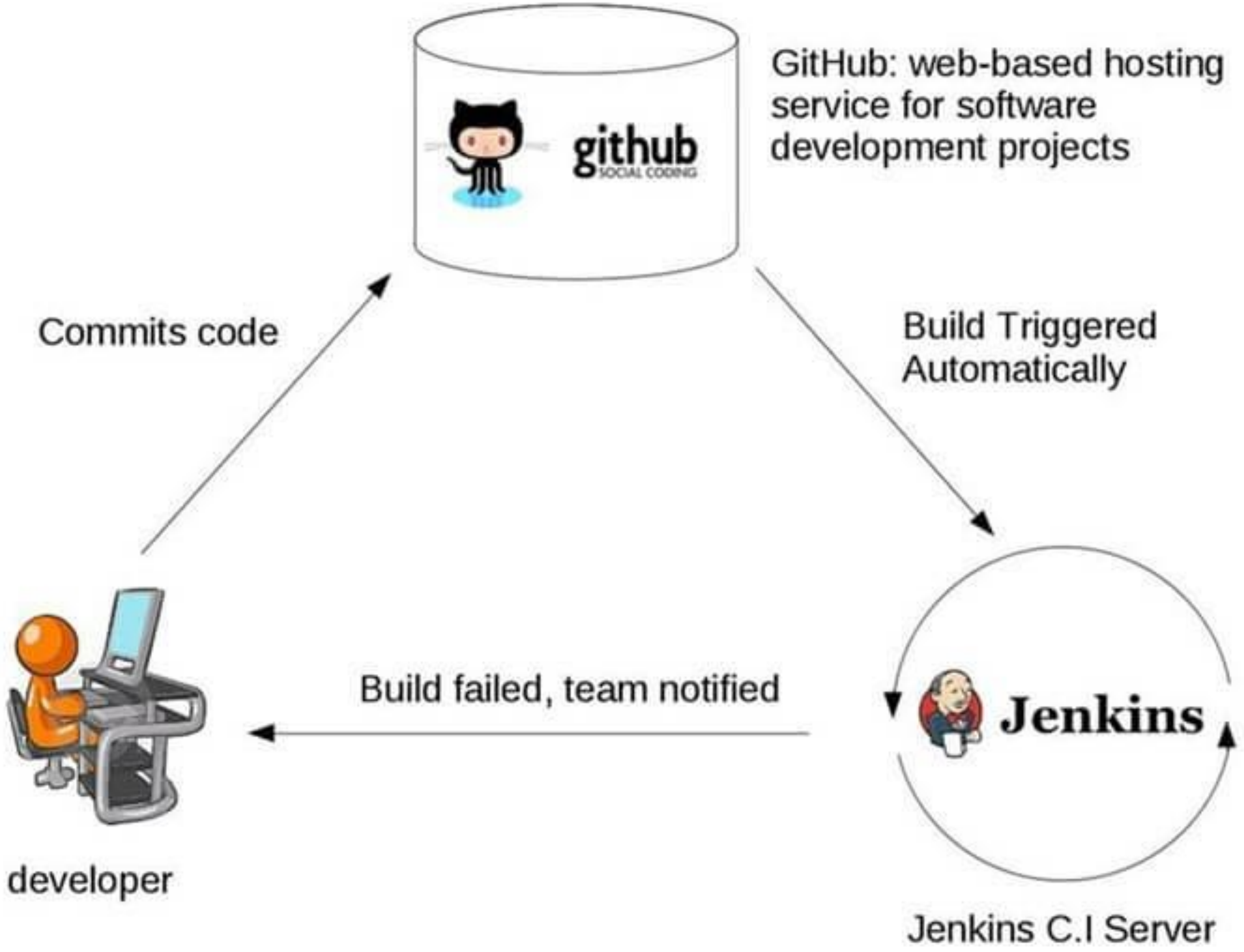
How does CI work?

- **Project build**

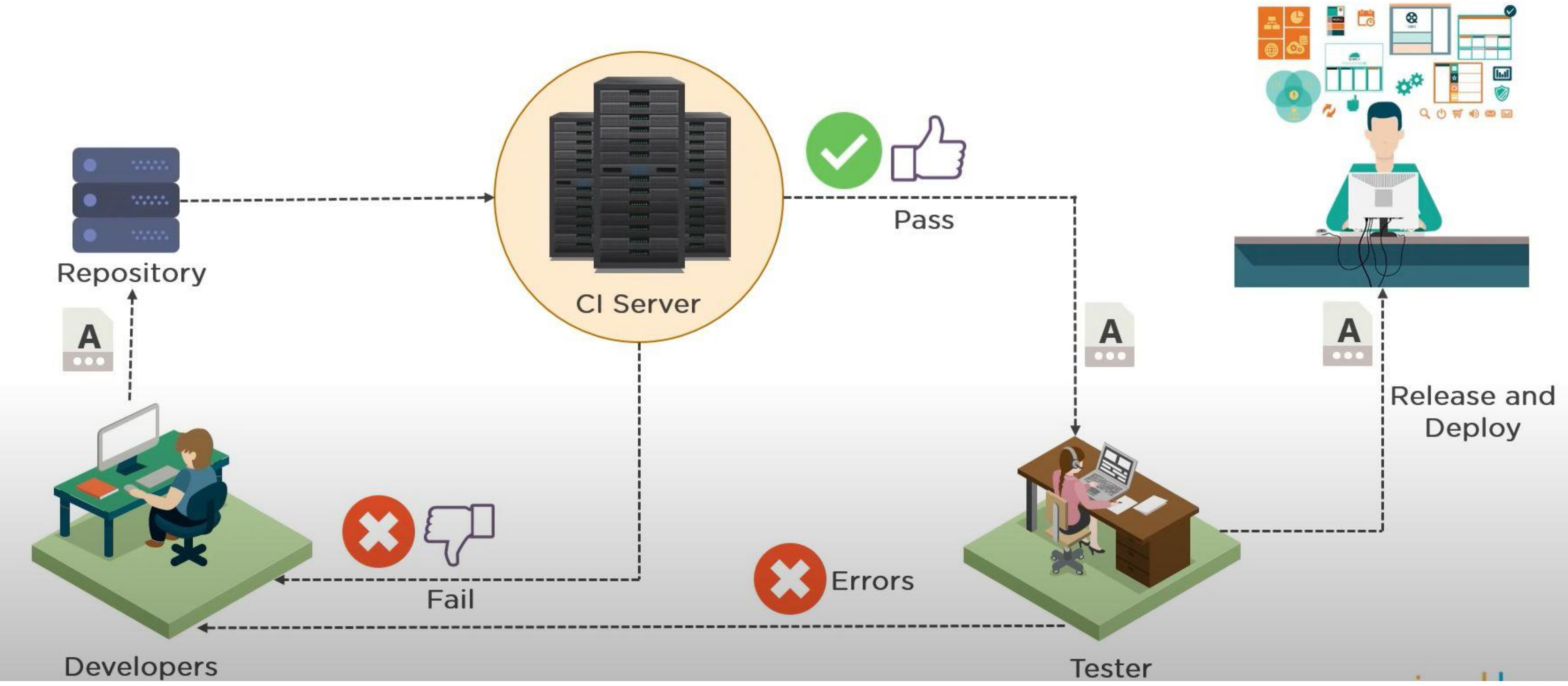
- **Nightly build**
  - After multiple commits from diverse developers during the day, the software is built every night.
    - Since the software is built only once in a day, it's a huge pain to isolate, identify, and fix the errors in a large codebase.

- **Continuous Integration approach**
  - The software is built and tested as soon as a developer commits code.
    - If any error is detected, the respective developer can quickly fix the defect.


Irregular commits
Developer 1
Developer 2
Developer 3
Code Repository
Issues in Integration
Delay in Testing


GitHub: web-based hosting service for software development projects
github SOCIAL CODING
Commits code
Build Triggered Automatically
Build failed, team notified
Jenkins
developer
Jenkins C.I Server

Continuous Integration



NetCICD - DevOps

Benefits of CI

- Allows you to maintain just a single source repository
- You can test the clone of the production CI environment
- The built environment should be close to the production environment
- One of the advantages of continuous integration is constant availability of a current build
  - ***Not all or nothing automation***
  - ***TTM***
- The complete process of build and testing and deployment should be visible to all the holders
- Helps you to build better quality software
- CI process helps to scale up headcount and delivery output of engineering teams

Benefits of CI (cont.)

- CI allows software developers to work independently on features in parallel
- Helps you to conduct repeatable testing
- Increase visibility enabling greater communication
  - ***All parties can see all revisions & who did it***
- Helps develop a potentially shippable product for fully automated build
- Helps you to reduce risks by making deployment faster and more predictable
  - ***Just adding small pieces***
- Immediate feedback when issue arrives - critical
- Avoid last-minute confusion at release date and timing

Best Practices of CI

- Commit Early and Commit Often
- Fix build failures immediately
- Act on metrics
- Build-in every target environment and create artifacts from every build
- The build of the software needs to be carried out so that it can be <u>automated</u>
- Do not depend on an IDE (use CLI)
- Build and test everything when it changes (automated)

Best Practices of CI

- The database schema counts as everything
  - ***Helps you to find out key metrics and track them visually***
- Check-in often and early
  - ***Similar to "saving" document***
- Stronger source code control
- Continuous integration is running unit tests whenever you commit code
- Keep the build fast with automated deployment

Unit Testing

What is Unit Testing?

- **Unit Testing (UT)**
  - A type of software testing where individual units or components of a software are tested.
    - ***A unit may be an individual function, method, procedure, module, or object.***
  - The purpose is to validate that each unit of the software code performs as expected.
  - Unit Testing is done during the development (coding phase) of an application by the developers.
  - Unit Tests isolate a section of code and verify its correctness.

Why Unit Testing?

- Unit tests help to fix bugs early in the development cycle and save costs.
- It helps the developers to understand the testing code base and enables them to make changes quickly
- Good unit tests serve as project documentation
- Unit tests help with code re-use. Migrate both your code and your tests to your new project. Tweak the code until the tests run again.
- Not only do you write your code; you write code to test your code!

How to do Unit Testing?

- **1. A developer writes a section of code in the application just to test the function.**
  - They would later comment out and finally remove the test code when the application is deployed.
  - We have done this in previous labs (automation 1)

- **2. A developer could also isolate the function to test it more rigorously.**
  - This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment.
  - Isolating the code helps in revealing unnecessary dependencies between the code being tested and other units or data spaces in the product. These dependencies can then be eliminated.
    - ***Now we do this in the lab***

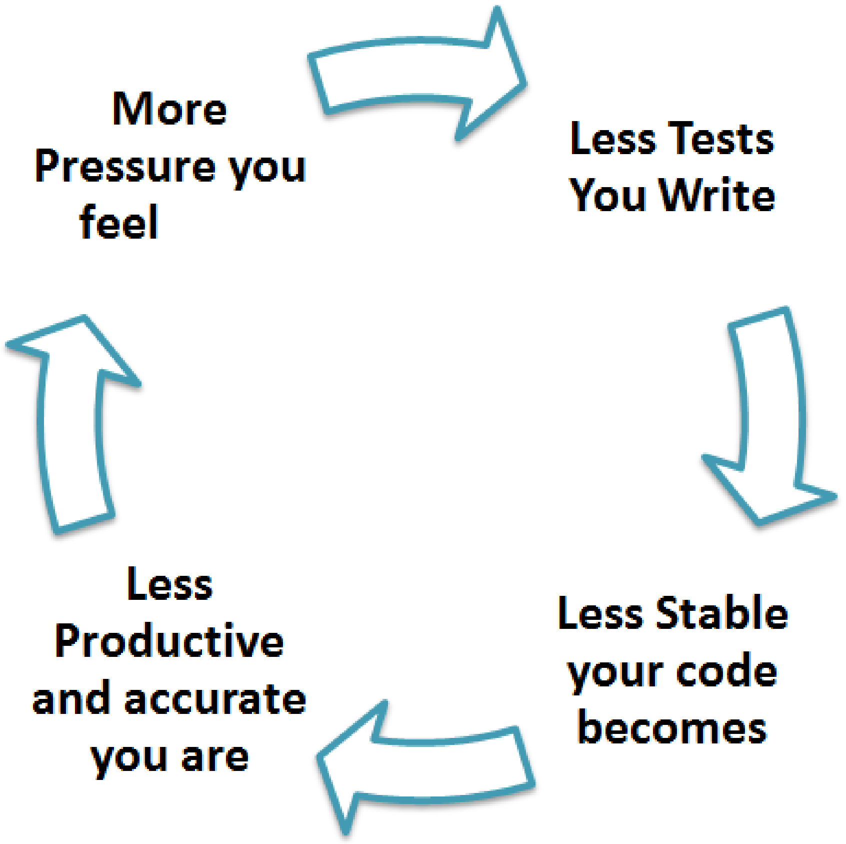
How to do Unit Testing? (cont.)

- **A coder generally uses a Unit Test Framework to develop automated test cases.**
  - Using an automation framework, the developer codes criteria into the test to verify the correctness of the code.
  - During execution of the test cases, the framework logs failing test cases.
    - ***Many frameworks will also automatically flag and report, in summary, these failed test cases.***
    - ***Depending on the severity of a failure, the framework may halt subsequent testing.***

- **The workflow of Unit Testing is:**
  - 1) Create Test Cases
  - 2) Review/Rework
  - 3) Baseline
  - 4) Execute Test Cases

The Unit Testing Myth

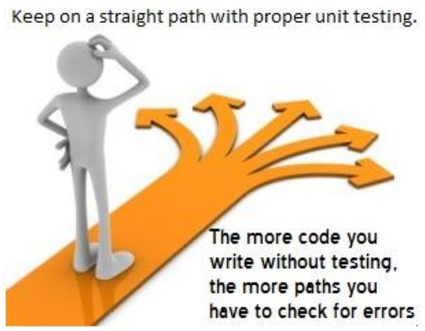
- **Myth**
  - "It requires time, and I am always overscheduled."
  - "My code is rock solid! I do not need unit tests."


More Pressure you feel
Less Tests You Write
Less Stable your code becomes
Less Productive and accurate you are

Advantages

- Developers looking to learn what functionality is provided by a unit and how to use it can look at the unit tests to gain a basic understanding of the unit API.
- Unit testing allows the programmer to refactor code later, and make sure the module still works correctly (i.e. Regression testing).
  - The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified and fixed.
- Due to the modular nature of the unit testing, we can test parts of the project without waiting for others to be completed.

Best Practices


Keep on a straight path with proper unit testing.
The more code you write without testing, the more paths you have to check for errors

- Unit Test cases should be independent
  - In case of any enhancements or change in requirements, unit test cases should not be affected
- Test only one code at a time
- Follow clear and consistent naming conventions for your unit tests
- In case of a change in code in any module, ensure there is a corresponding unit test case for the module, and the module passes the tests before changing the implementation
- Bugs identified during unit testing must be fixed before proceeding to the next phase in SDLC
- Adopt a “test as your code” approach
  - The more code you write without testing, the more paths you have to check for errors

How do we do CI/CD and UT?

What is Jenkins?

- Jenkins is an open-source CI server for orchestrating a chain of actions to achieve the CI process in an automated fashion

- Jenkins supports the complete development life cycle of software
  - ***Building, testing, documenting the software, deploying***

Who uses Jenkins?

DELL

ebay

Fidelity INVESTMENTS

SONY

github SOCIAL CODING

f

Etsy

NASA

NETFLIX

salesforce

tumblr.

intuit

LinkedIn

YAHOO!

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.
Since the code was built all at once, some developers would need to wait until other developers finish coding to check their build	The code is built immediately after any of the Developer commits.
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.

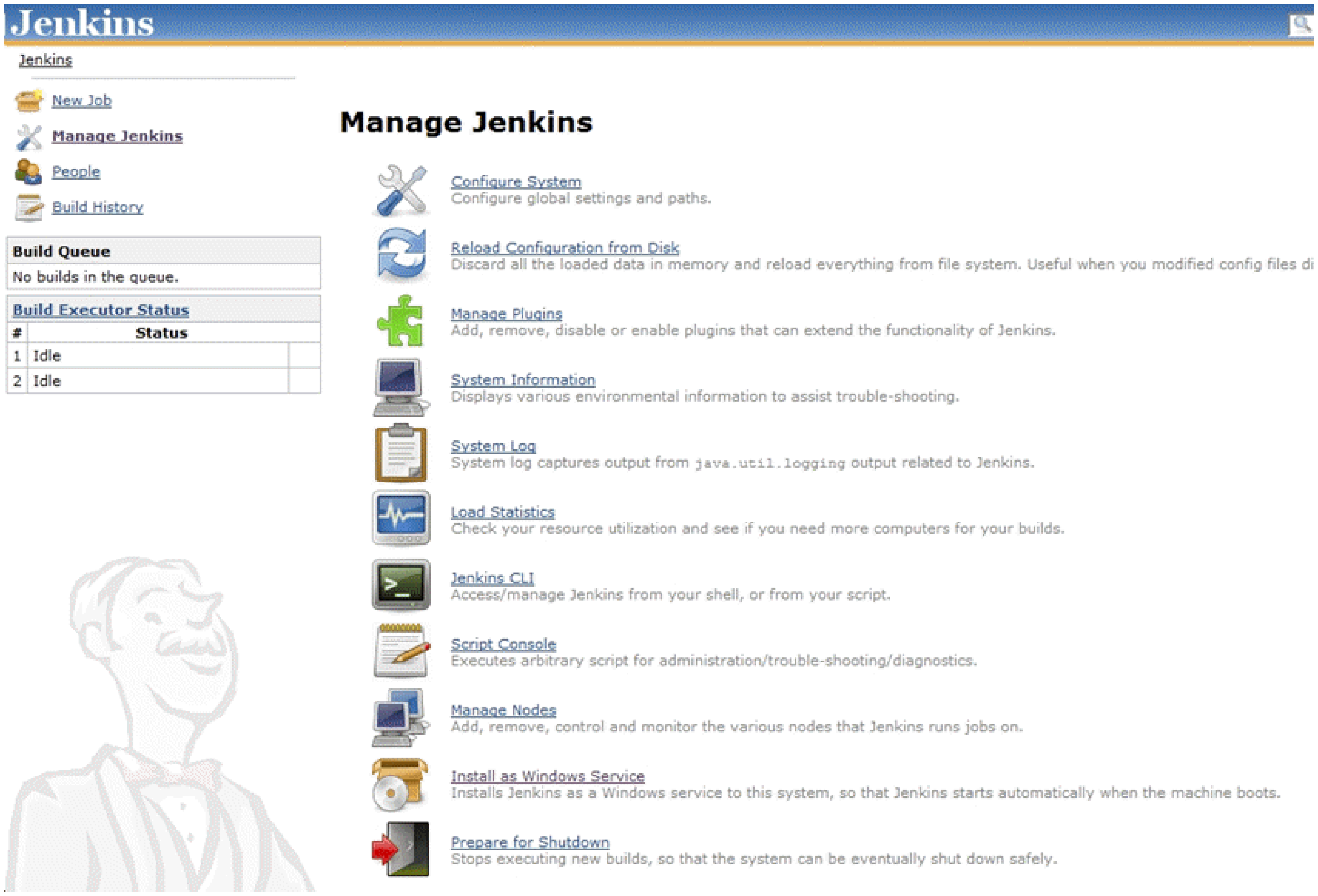
Jenkins for a Developer

- **Easy to install**
  - Download one file – jenkins.war
  - Run one command – java –jar jenkins.war

- **Easy to use**
  - Create a new job – checkout and build a small project
  - Check in a change – watch it build
  - Create a test – watch it build and run
  - Fix a test – check in and watch it pass

- **Multi-technology**
  - Build C, Java, C#, Python, Perl, SQL, etc.
  - Test with Junit, Nunit, MSTest, etc.
  - Also AWS, GNS3, Ansible

Jenkins Plugins



Jenkins

Jenkins

New Job

Manage Jenkins

People

Build History

Build Queue

No builds in the queue.

Build Executor Status

#	Status
1	Idle
2	Idle

Manage Jenkins

Configure System
Configure global settings and paths.

Reload Configuration from Disk
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files di

Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

System Information
Displays various environmental information to assist trouble-shooting.

System Log
System log captures output from `java.util.logging` output related to Jenkins.

Load Statistics
Check your resource utilization and see if you need more computers for your builds.

Jenkins CLI
Access/manage Jenkins from your shell, or from your script.

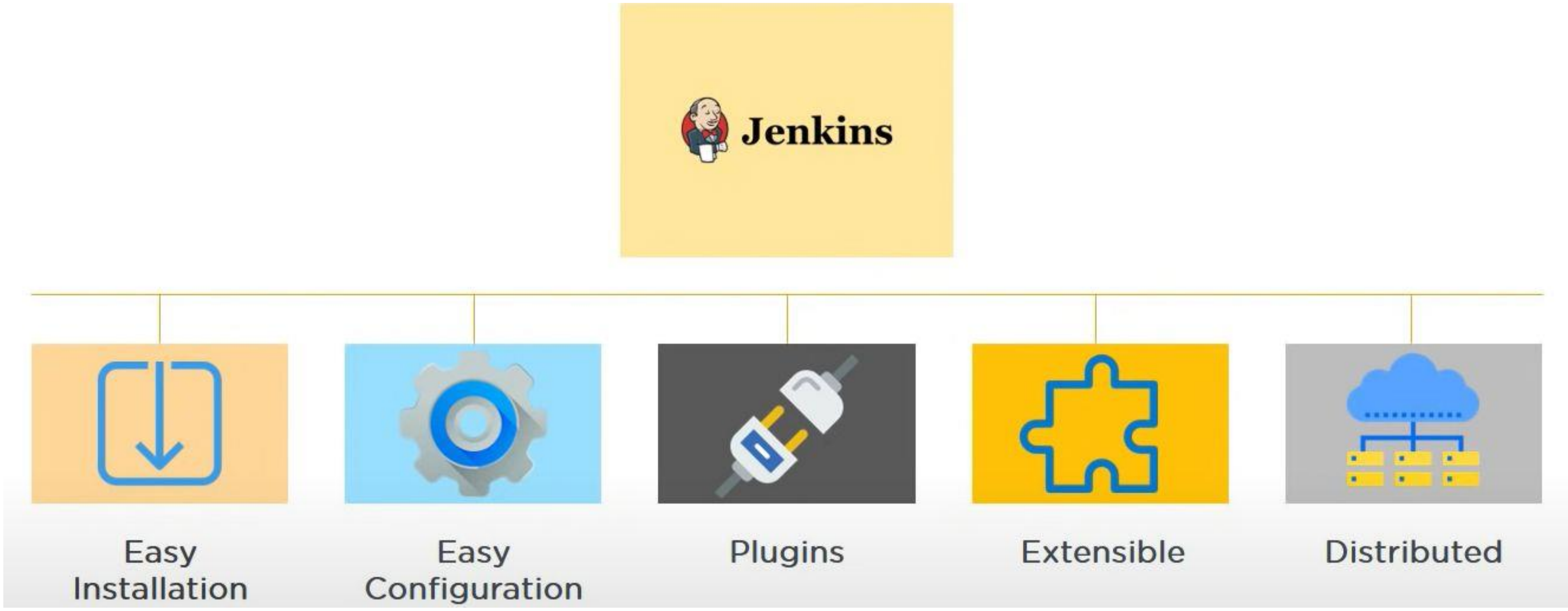
Script Console
Executes arbitrary script for administration/trouble-shooting/diagnostics.

Manage Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

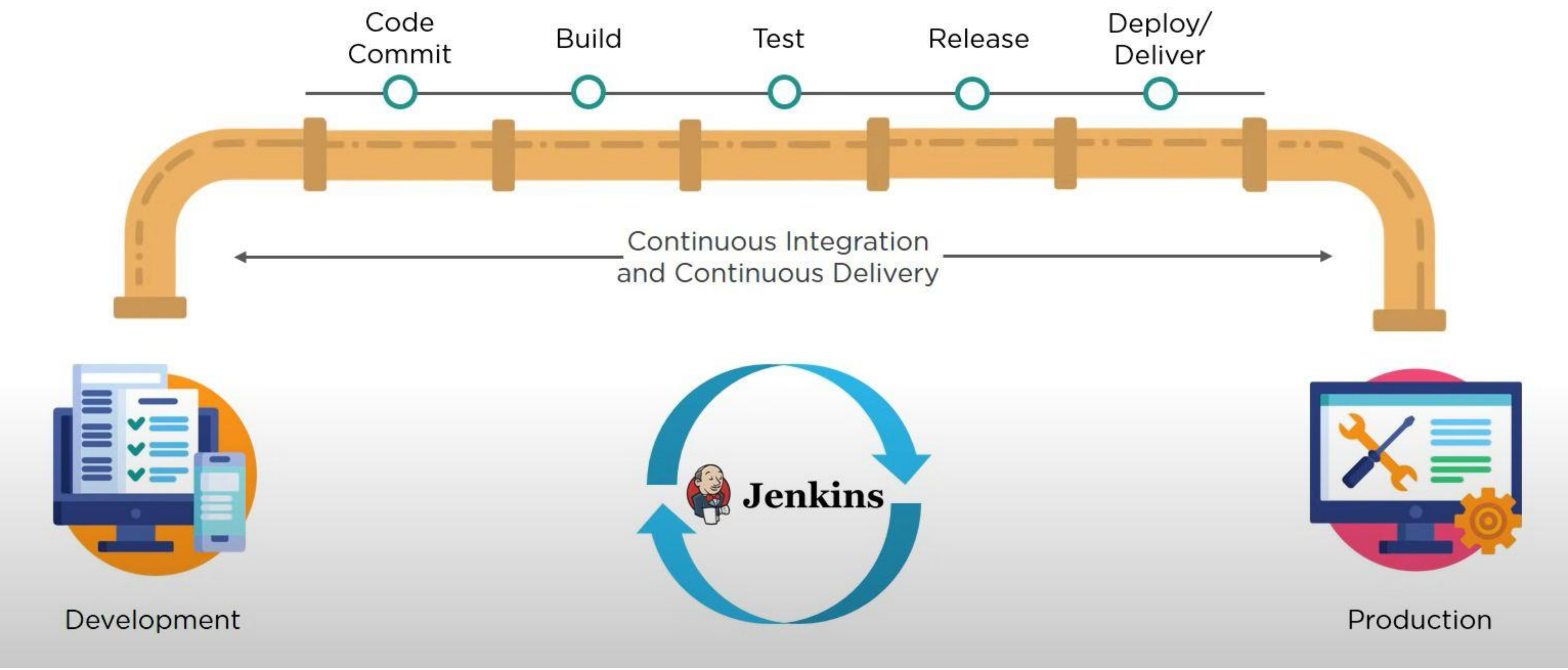
Install as Windows Service
Installs Jenkins as a Windows service to this system, so that Jenkins starts automatically when the machine boots.

Prepare for Shutdown
Stops executing new builds, so that the system can be eventually shut down safely.

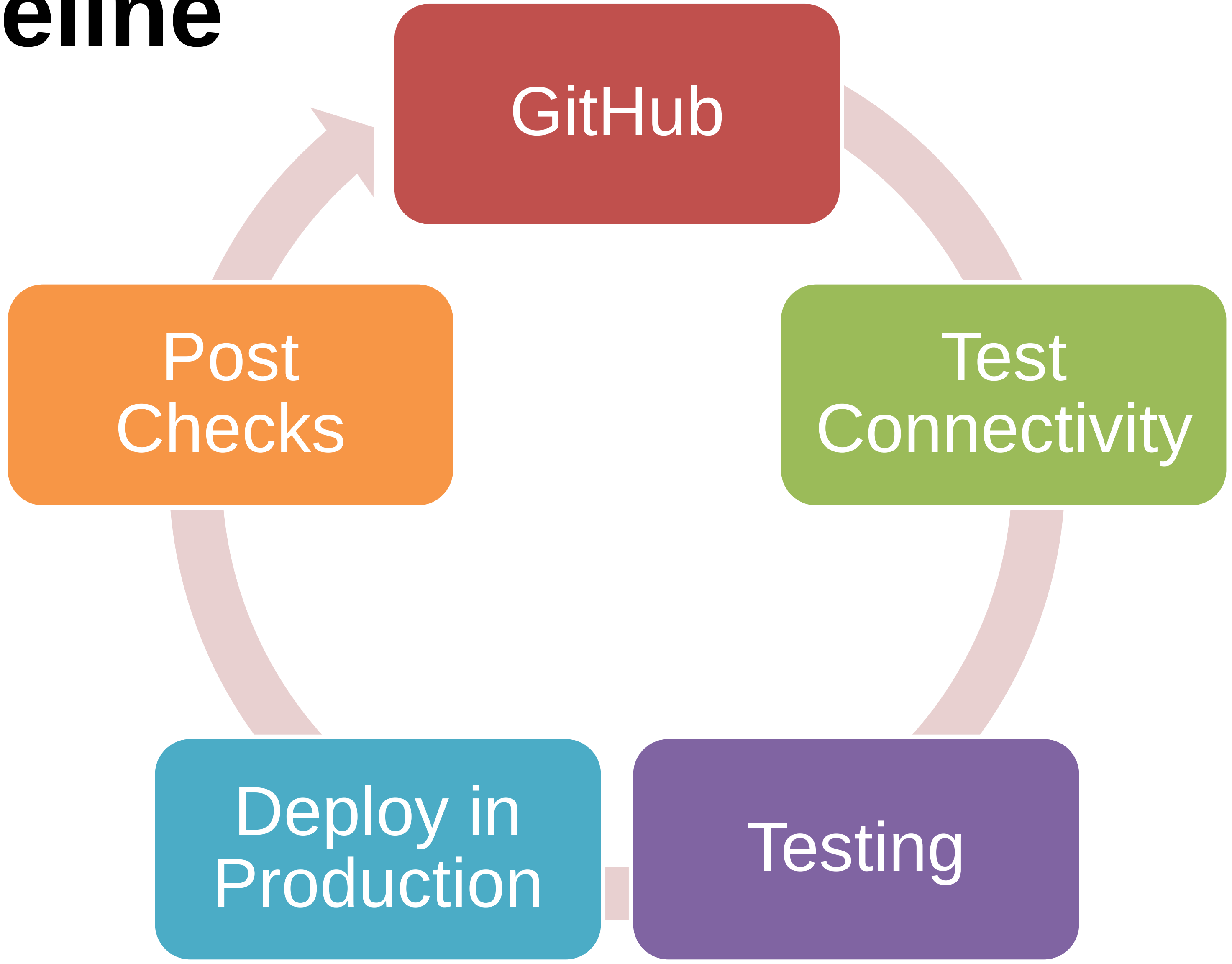
Jenkins Features



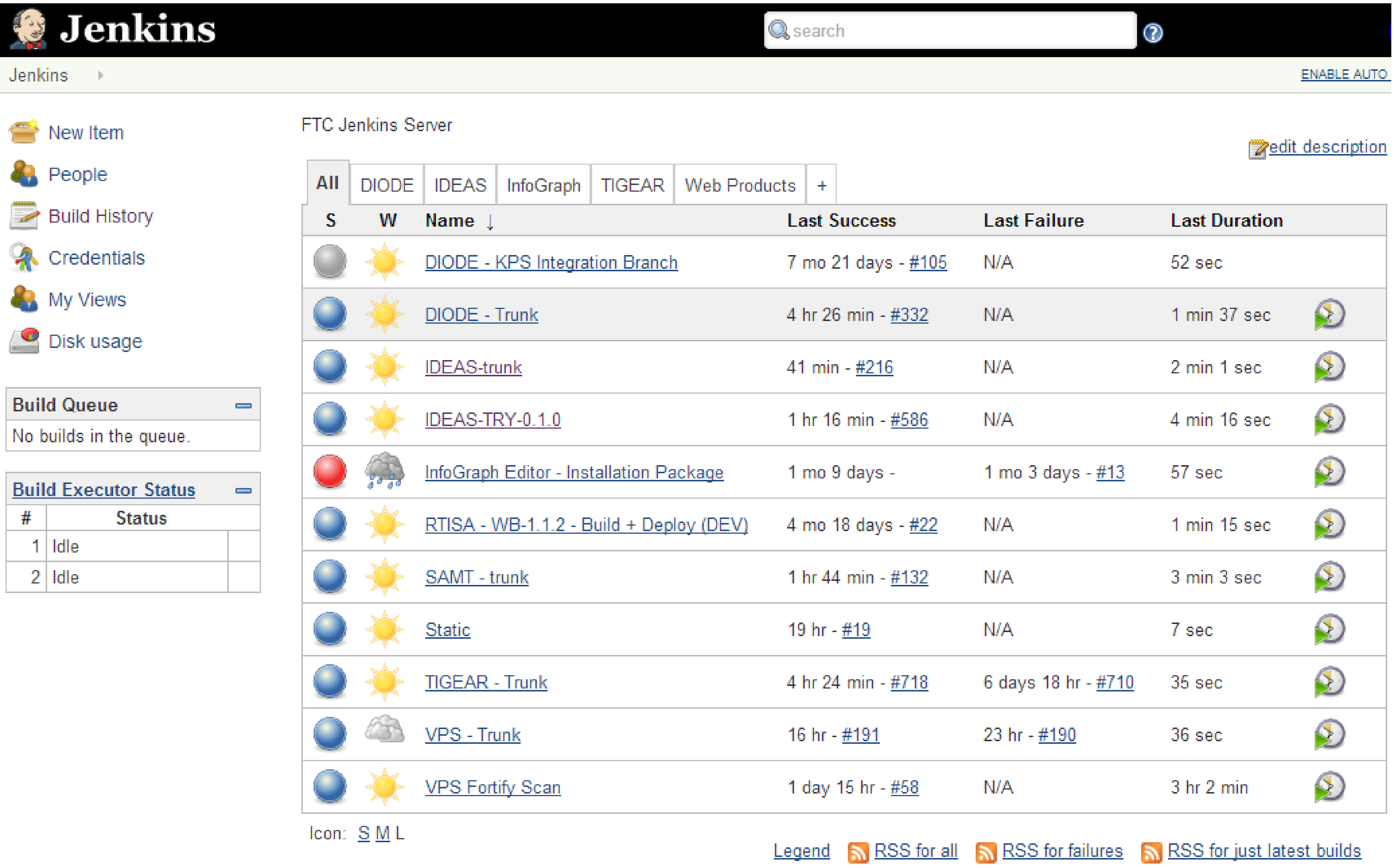
Jenkins Pipeline



Pipeline


GitHub
Test Connectivity
Testing
Deploy in Production
Post Checks

Dashboard



FTC Jenkins Server

Build Queue

No builds in the queue.

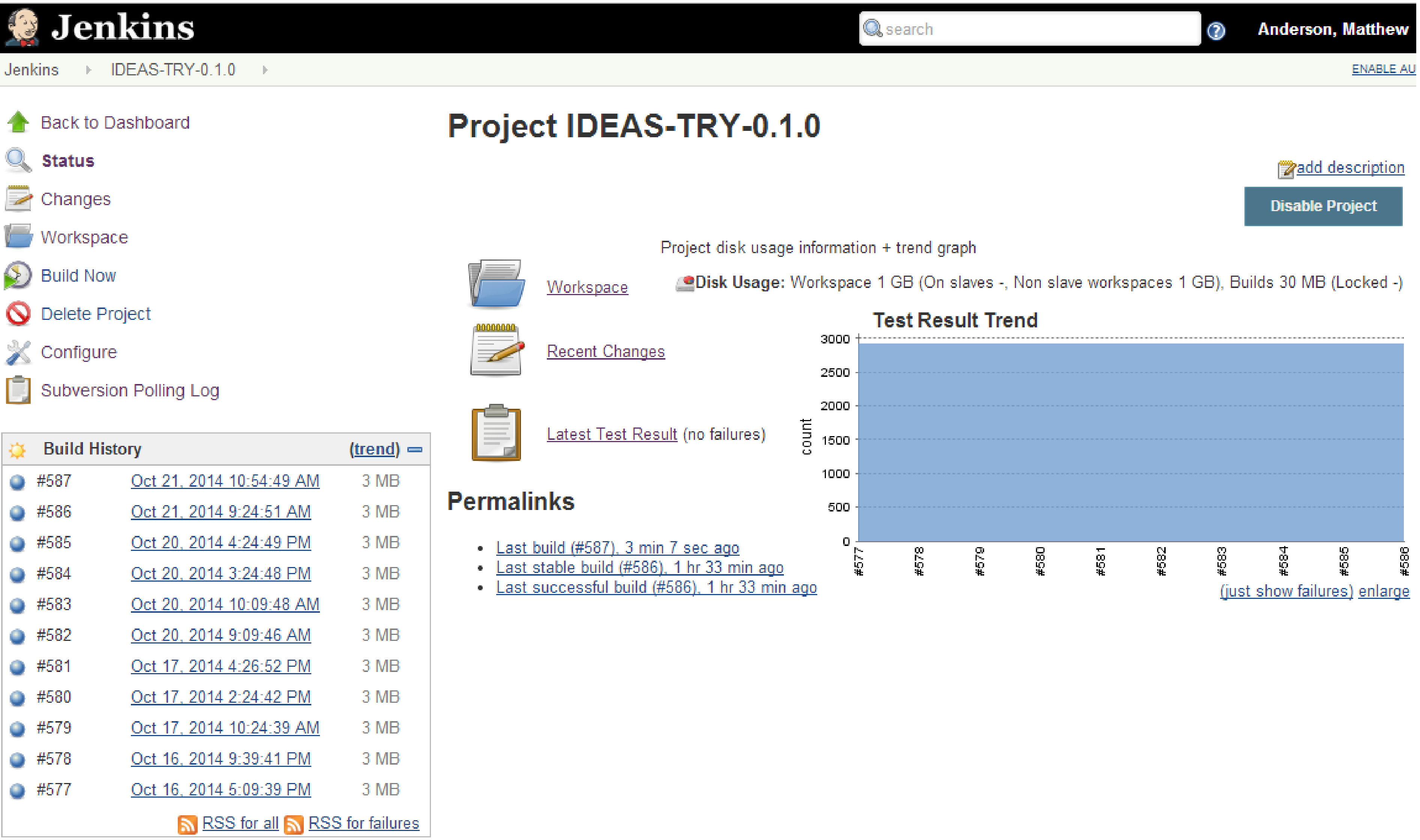
Build Executor Status

#	Status
1	Idle
2	Idle

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

Project View



Project IDEAS-TRY-0.1.0

Project disk usage information + trend graph

Workspace

Disk Usage: Workspace 1 GB (On slaves -, Non slave workspaces 1 GB), Builds 30 MB (Locked -)

Recent Changes

Latest Test Result (no failures)

Test Result Trend

Permalinks

- Last build (#587), 3 min 7 sec ago
- Last stable build (#586), 1 hr 33 min ago
- Last successful build (#586), 1 hr 33 min ago

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
#577	Oct 16, 2014 5:09:39 PM	3 MB

RSS for all RSS for failures

Questions?