TRAVIS ROGERS, PHOENIX

# DEVOPS PIPELINES OVERVIEW

# TONIGHT'S AGENDA

SPEAKER BACKGROUND

AGILE & DEVOPS

DEVOPS PIPELINE EXAMPLES

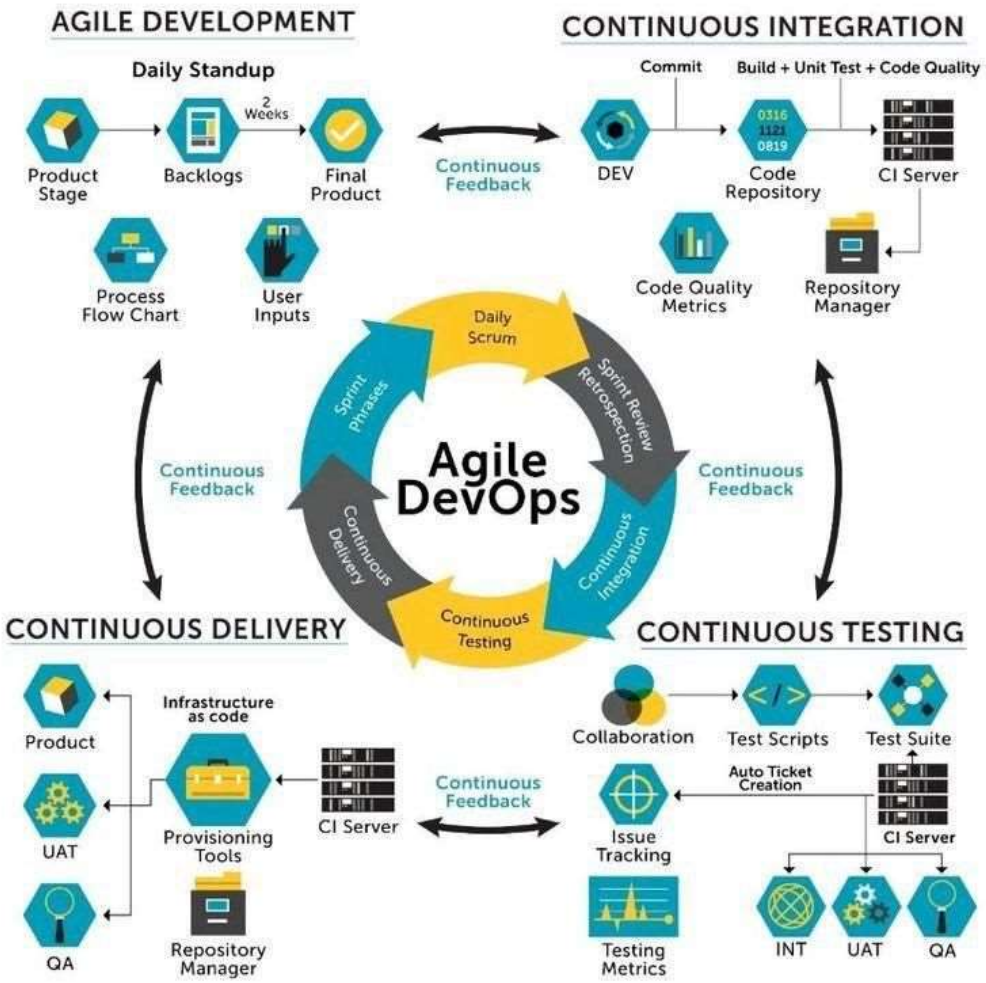
STEPS FOR DEFINING YOUR DEVOPS PIPELINE

AZURE DEMO

DISCUSSION

## Agile with Scrum + DevOps

Scrum focuses on releasing production software at the end of sprints. DevOps enables this process through Continuous Integration, Continuous Testing, and Continuous Delivery. Full automation should be the goal.

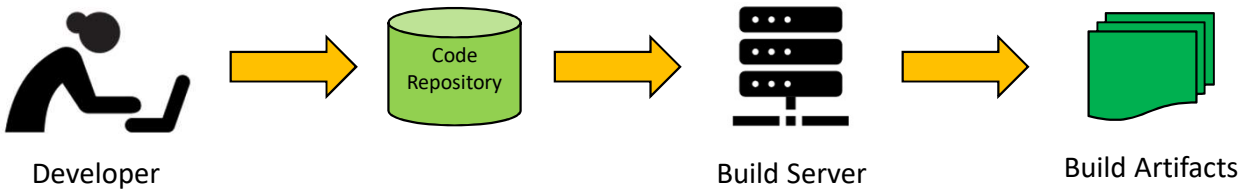


# DEVOPS PIPELINE EXAMPLES

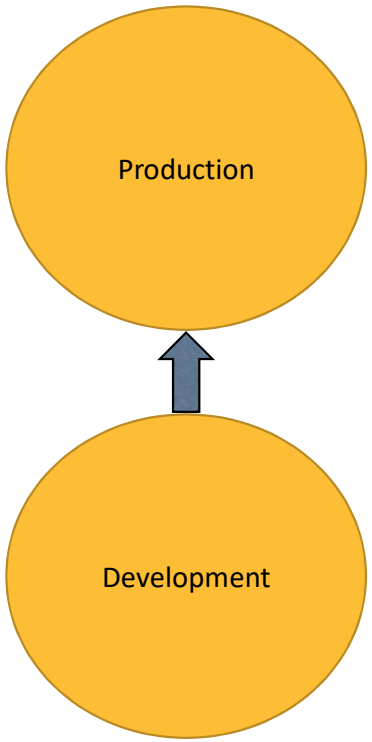
# The Precursor

- All DevOps examples include source code repositories
- The starting point for the examples is after a developer checks in code
- Infrastructure & Application Architecture is mostly omitted for this discussion

*Simplified Build Process*

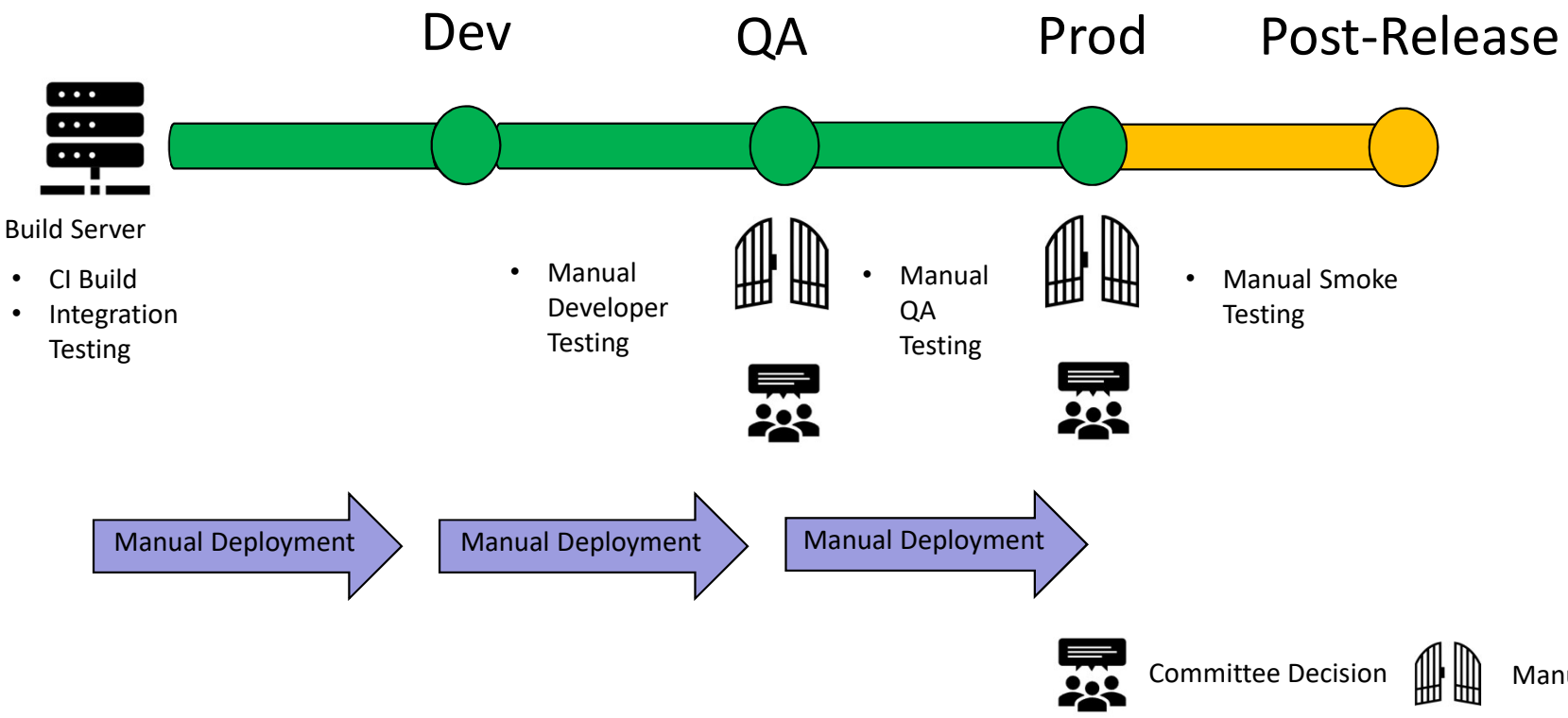


# The First DevOps Pipeline



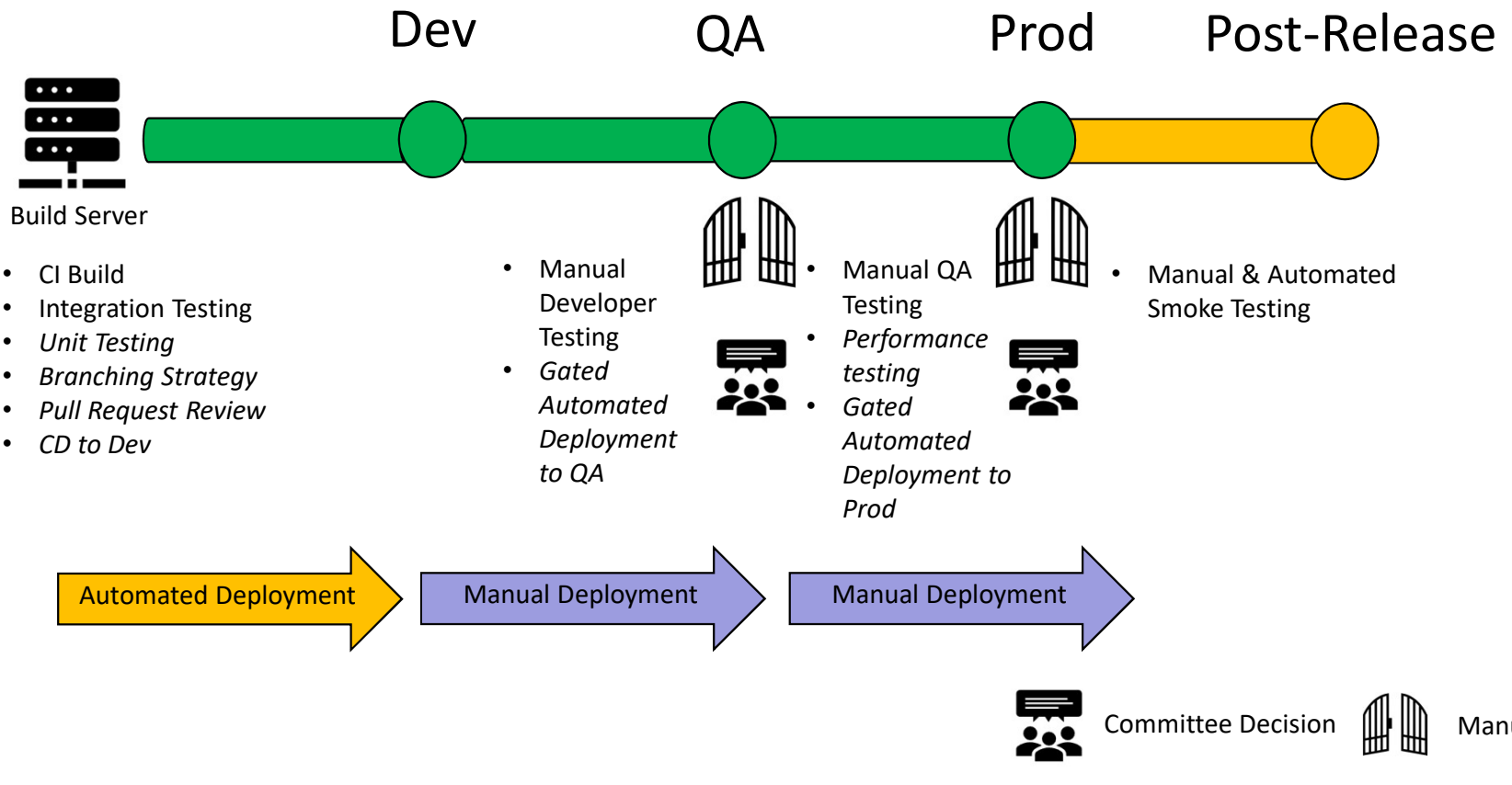
# Example DevOps Pipeline

*Starter*



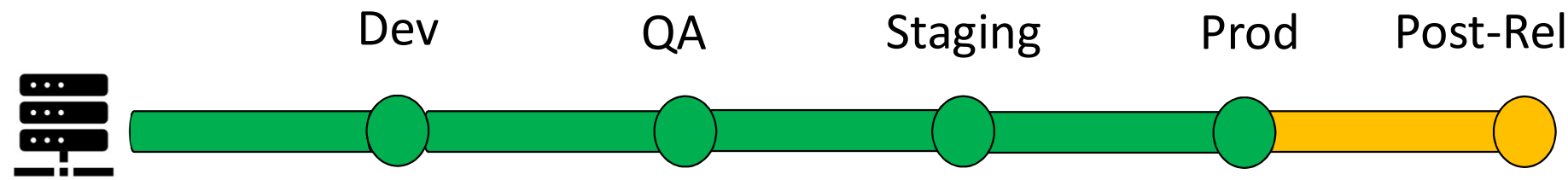
# Example DevOps Pipeline

*Mature*



# Example DevOps Pipeline

*Advanced*




Build Server

- CI Build
- Unit Testing
- Integration Testing
- Pull Request Review
- *Code Coverage Check*
- *Advanced Branching Strategy*
- *Code Style Checks*
- *Security Checks*
- *CD Deployment to Dev & QA*

- *Ad-hoc manual Developer Testing*

- *Automated Environment Creation*
- *Automated Database deployment seeding*
- *Automated API Testing*
- *Automated UI Testing*
- *Automated 3rd Party Integration Testing*
- *CD Deployment to Staging*

- *Automated Environment Creation*
- *Automated Smoke Tests*
- *Automated Performance Testing*
- *Automated Security Testing*
- *CD Deployment to Production*

- *Automated UI & API Smoke Tests*
- *Automated Security Checks*

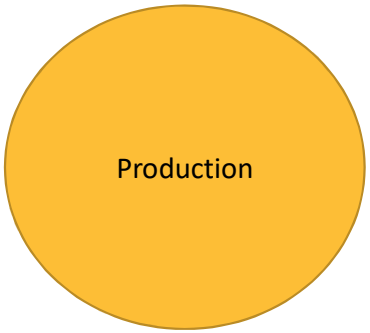
Automated Deployment & Testing

Committee Decision

Manual Gates

# Example DevOps Pipeline

*The Future*
*(Possible)*



# Example DevOps Pipeline

*Detailed Containerized (step-by-step)*

**Make the changes**
•Create a new branch in git
•Make the changes behind a feature flag
•Run unit tests to validate your changes with the feature flag both on and off
**Pull request**
•Commit the changes
•Push the changes to a remote on github
•Make a pull request
•CI build runs automatically in the background
•Code review
•Repeat this step a few times perhaps
•Merge the changes into git master
**CI runs on master**
•Install frontend dependencies via npm
•Build/optimize HTML+CSS+JS assets
•Run frontend unit/functional tests
•Install Python dependencies from PyPI
•Run backend unit/functional tests
•Run integration tests against both assets
•Push frontend assets to a CDN
•Build a container for the Python program
•Push container to registry
•Update kubernetes manifest

# Example DevOps Pipeline

## *Detailed Containerized (step-by-step) continued*

**Replace old code with new code**

•Kubernetes spins up some instances of the new container
•Kubernetes waits for those instances to become healthy
•Kubernetes add those instances to the HTTP load balancer
•Kubernetes waits for old instances to become unused
•Kubernetes spins down old instances
•Kubernetes repeats until all old instances have been replaced with new ones

**Enable new feature flag**

•Enable the new code for just yourself, gain confidence
•Enable the new code for 10% of your users, watch operational and business metrics
•Enable the new code for 50% of your users, watch operational and business metrics
•Enable the new code for 100% of your users, watch operational and business metrics
•Finally, go through the entire process again to remove the old code and the feature flag

# PIPELINE RELEASE TIMES

| NAME | TASKS | GATES | RELEASE TIME |
|---|---|---|---|
| Starter | 5 | 2 | Days/Weeks/Months |
| Mature | 12 | 2 | Days/Weeks/Months |
| Advanced | 23 | 0 | Minutes/Hours |
| Detailed Containerized | 32 | 4 | Minutes/Hours |
| The Future (Possible) | 1 | 0 | 50 ms |

# FUTURE OF DEVOPS

*Continuous Deployment and simplified pipelines are key.*

## CODE REPOSITORIES

- A single code repository
- Code Branches do not exist
- Feature Flags identify new Code
- Real-time collaboration like Google Docs

## BUILDS & ENVIRONMENTS

- The concept of builds does not exist
- Feature Flags act as Sandbox environments

## TESTING

- Unit tests are run in the background as you type your code

## RELEASES

- Releases occur after completing working code in a structured editor moving straight to production
- They are “deployless”



Source: *How Dark deploys code in 50ms*

# FOUR STEPS TO DEFINE YOUR DEVOPS PIPELINE

## A FEW NOTES

These are the steps I go through each time I am asked to define a DevOps pipeline. The goal is to reach full automation, but not every company is ready for it. Strive to build the best solution for the given environment.

1. ANALYZE THE CURRENT ARCHITECTURE
2. DEVOPS MATURITY
3. DETERMINE DEVOPS PIPELINE
4. TOOLING

# AZURE DEMO

# THANK YOU

## HOW TO CONTACT WITH ME

Don’t hesitate to contact with me.
I will be happy to answer your questions.


EMAIL: TRAVIS.ROGERS@NEUDESIC.COM

TWITTER: @TRAVISOFTHEWEB

LINKEDIN.COM/IN/TRAVIS-ROGERS-NET