Automating Code Security Testing

Peter Mosmans
LEAD PENETRATION TESTER

@onwebsecurity https://www.onwebsecurity.com

Scenario

Maeve

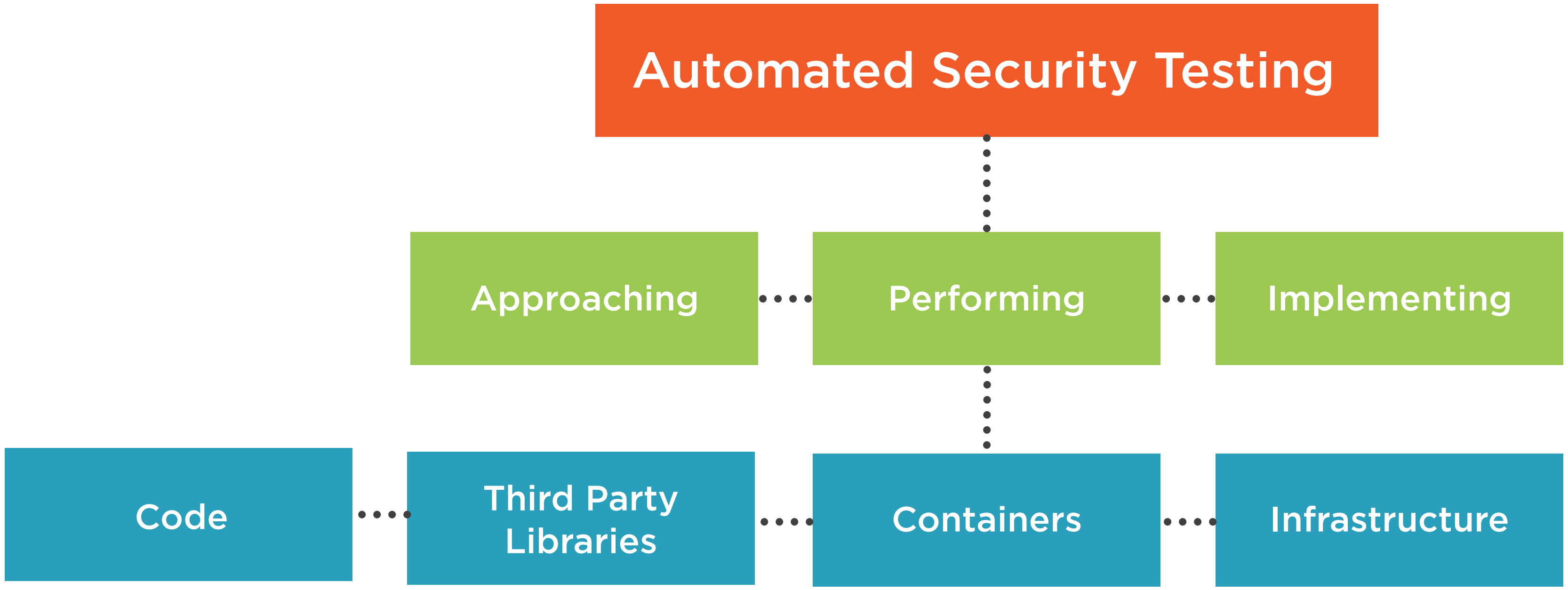
"Where should we start when performing tests?"

Jennifer

"Let's start with testing our own code"

"Using the right tools can make a lot of difference"

Automated Security Testing Overview



Automated Security Testing

Approaching

Performing

Implementing

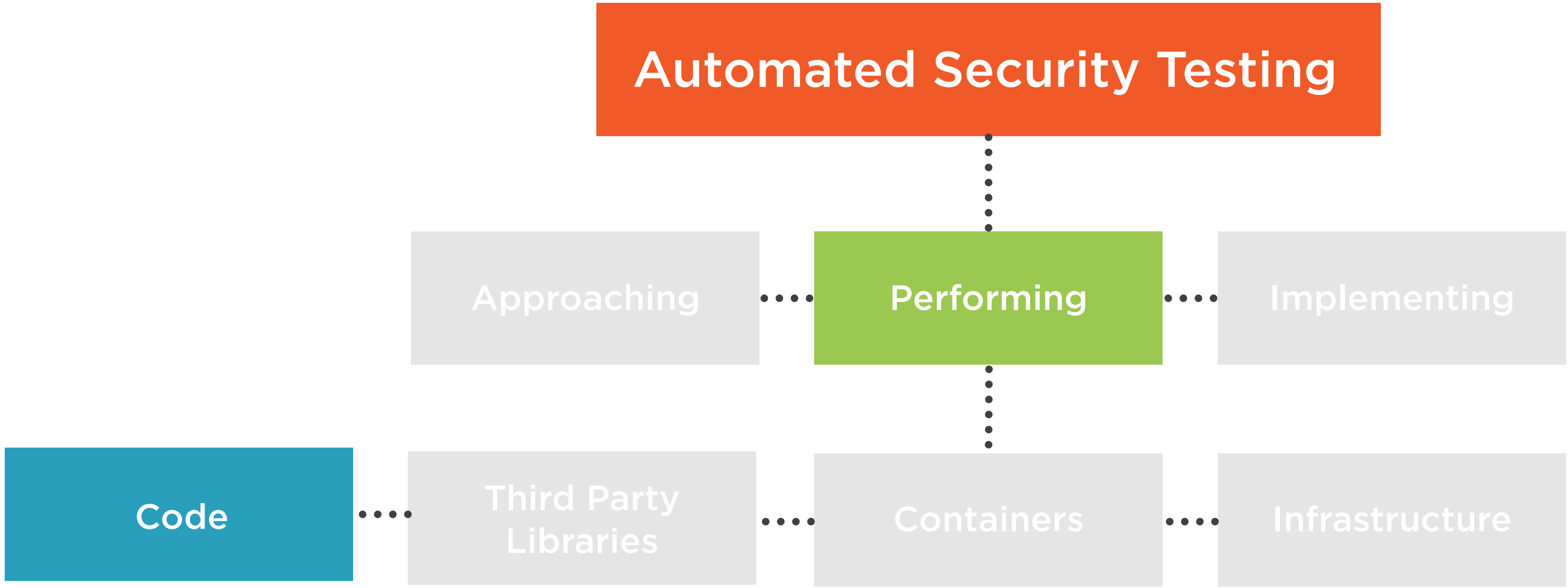
Code

Third Party Libraries

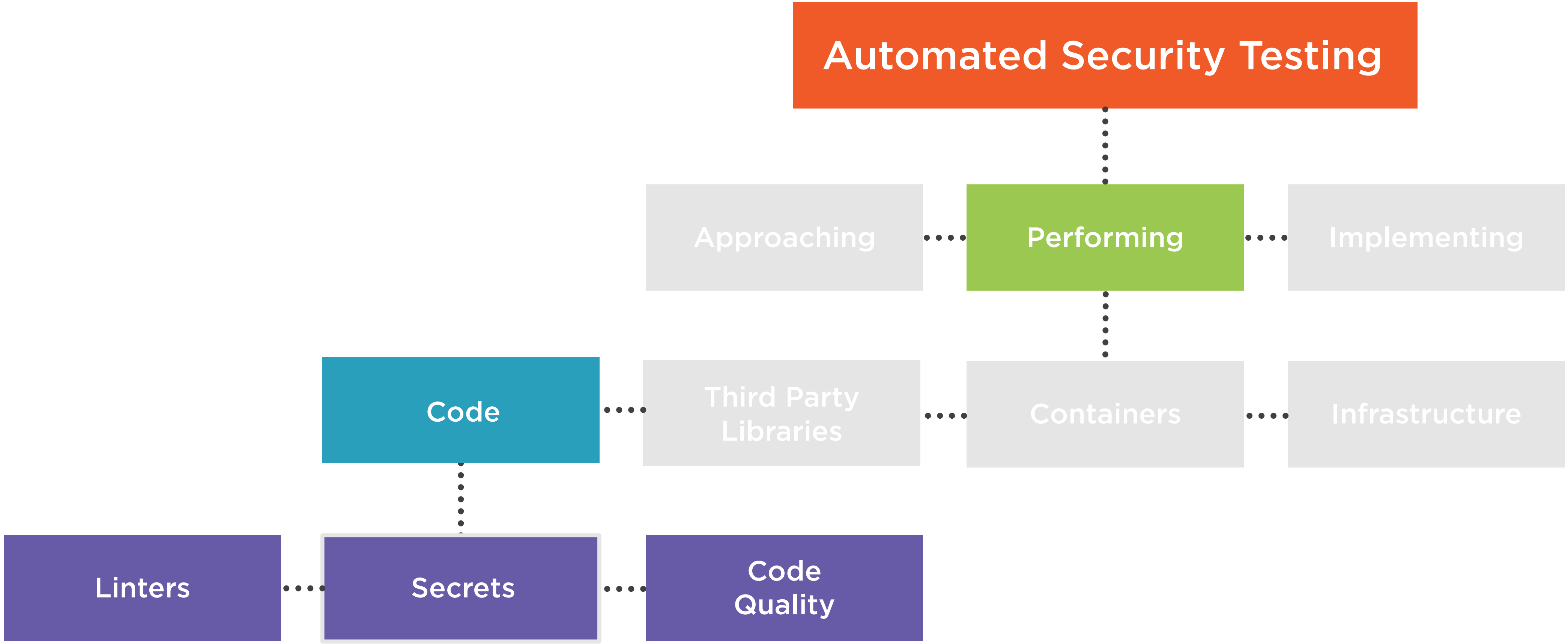
Containers

Infrastructure

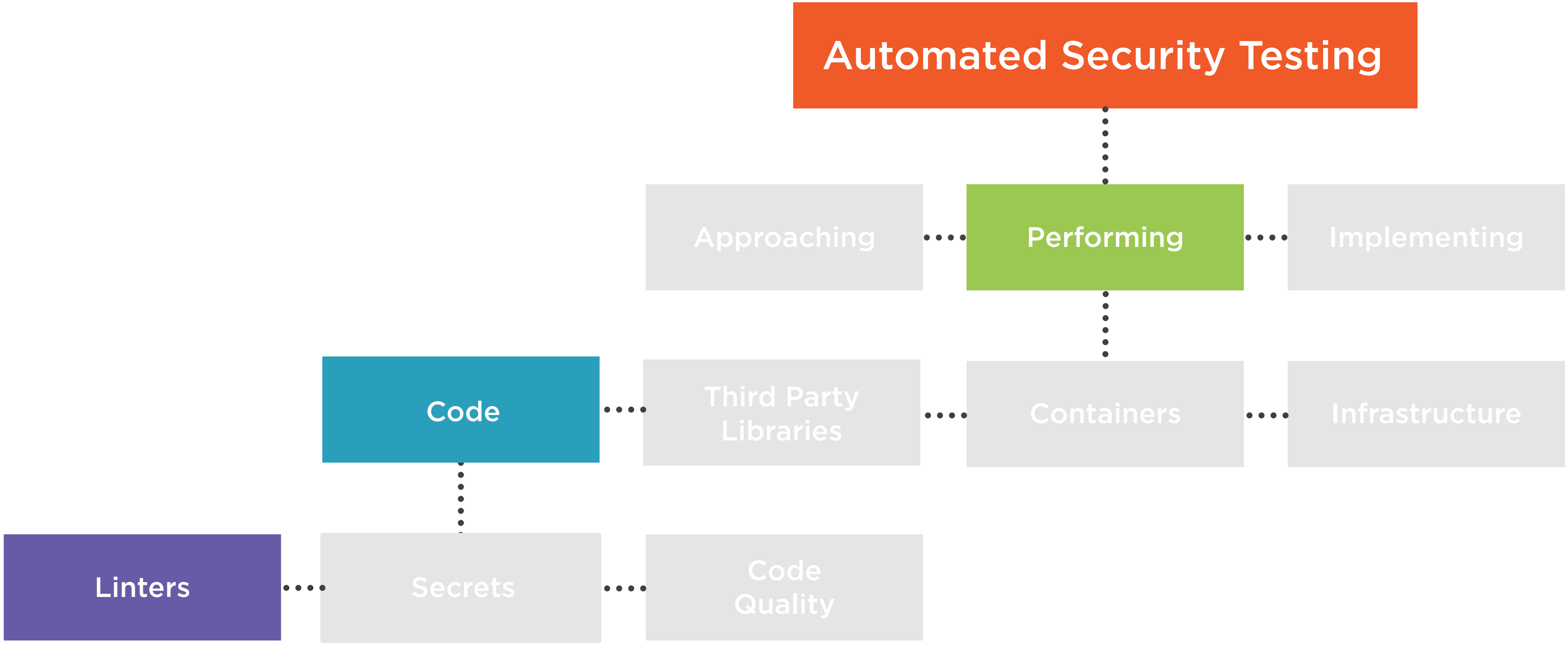
Automated Security Testing Overview



Automated Security Testing Overview



Automated Security Testing Overview



Module Overview

Using linters

Demo:

- Using a linter

Detecting secrets

Demos:

- Detecting existing secrets
- Using pre-commit hooks
- Detecting secrets in a pipeline

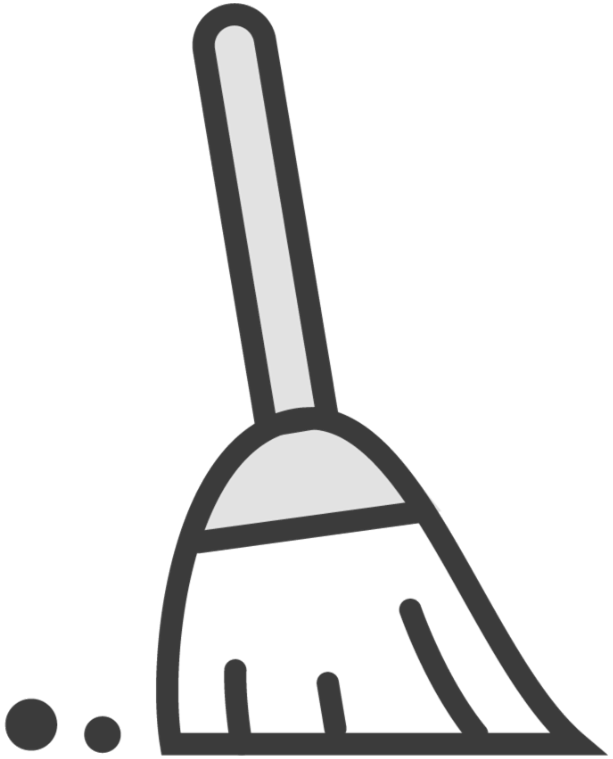
Using code quality metrics systems

Demos:

- Installing and using a code quality metrics system

Linting Code

What Can Linting Do?



Detect errors

Detect formatting or styling issues

Suggest best practices

Increases overall quality of the code

Makes maintenance of code easier

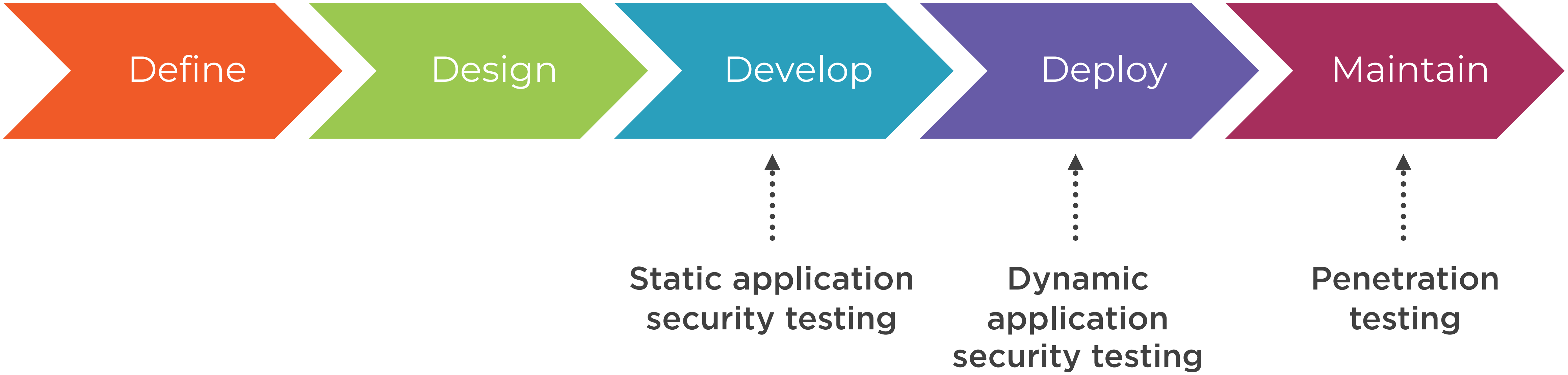
Issues With Linters

Not every language has “quality” standard linter tools available

Different versions or configurations can lead to different results

Information overload can lead to focusing on “unimportant” issues

Shift Left

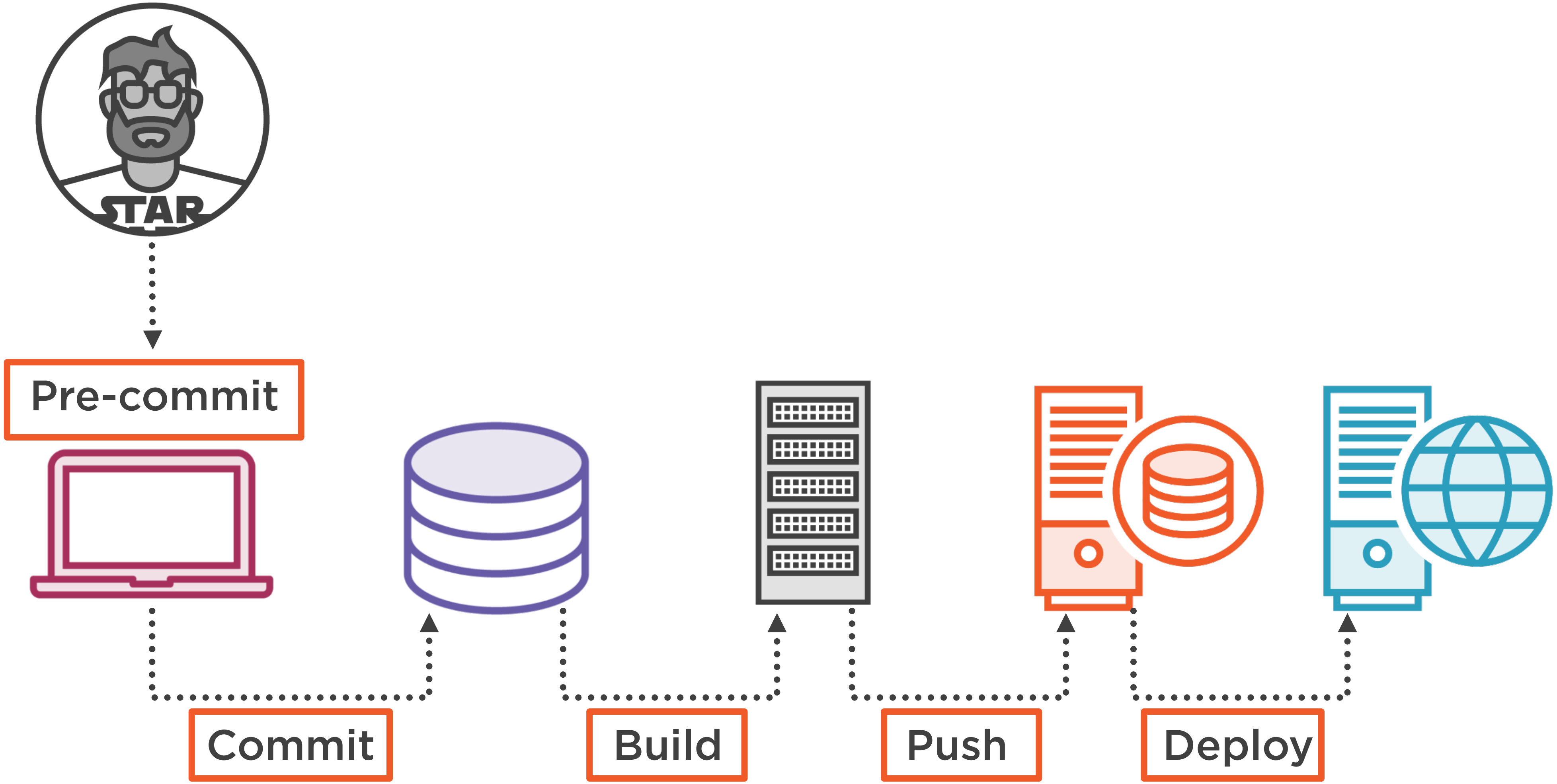


Static application security testing

Dynamic application security testing

Penetration testing

Where Security Tests Can Be Performed



Pre-commit

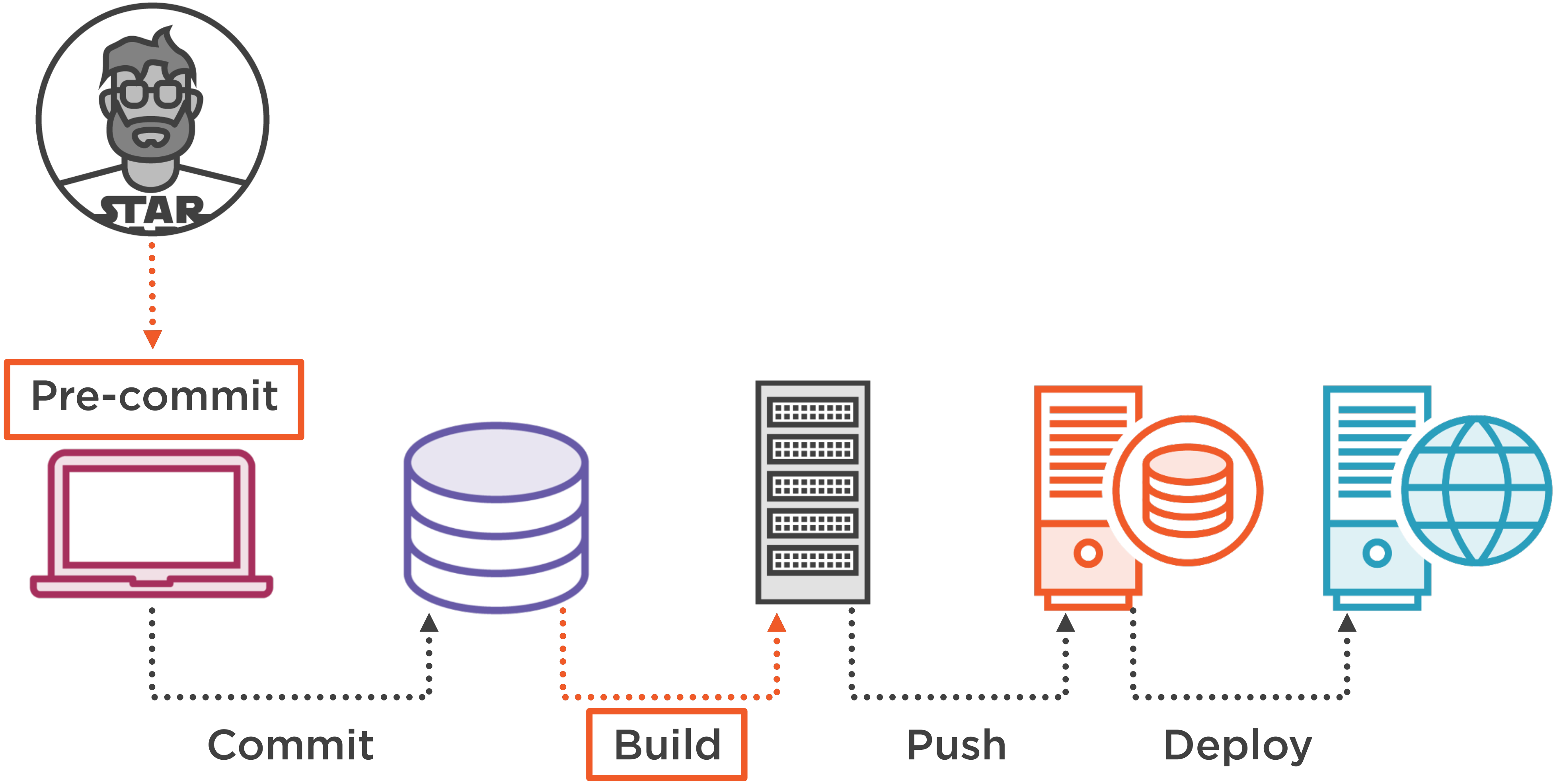
Commit

Build

Push

Deploy

Where and When to Use Linters



Pre-commit

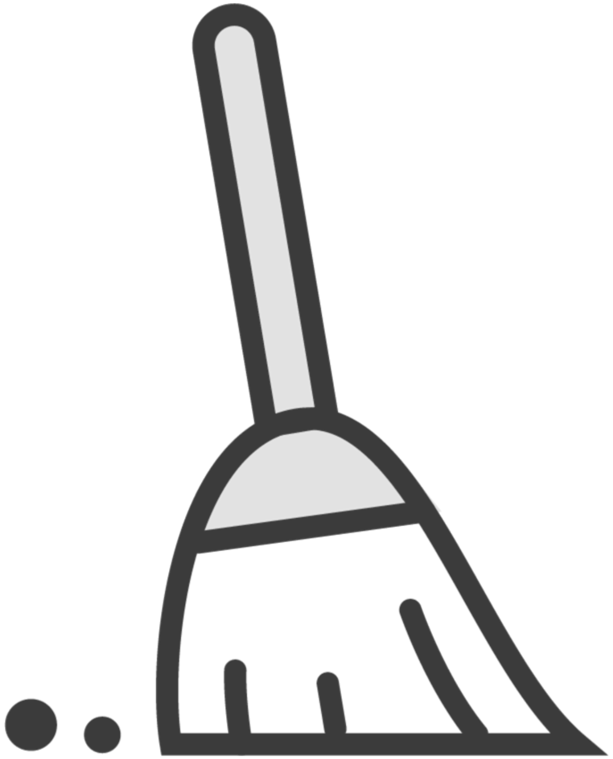
Commit

Build

Push

Deploy

Tool That Will Be Demo-ed



Haskell Dockerfile Linter (hadolint)

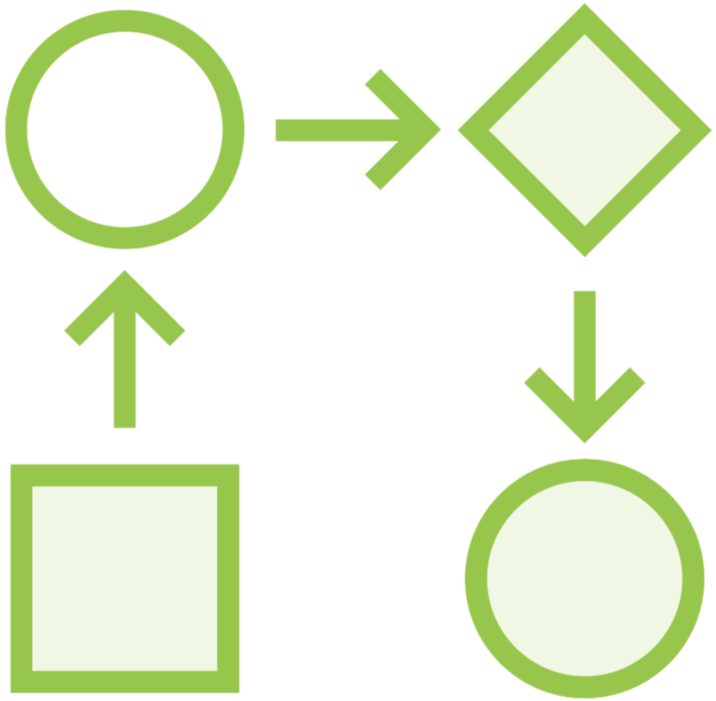
- Dockerfile linter enforcing best practices

Demo

Linting a Dockerfile

- Use linter on command-line
- Use linter in build pipeline
- Modify Jenkins job to conditionally push Docker image

Workflow for Linters



Agree upon tooling

- Create list of current linter issues
- Audit the list of issues
  - Is it a false positive or an issue?

Add configuration file to repository

Use configuration with every scan

- Warn or fail build for new issues

Update configuration when necessary

Linters

Detect errors

Detect formatting or styling issues

Suggest best practices

Advantage

Improves readability

Improves consistency

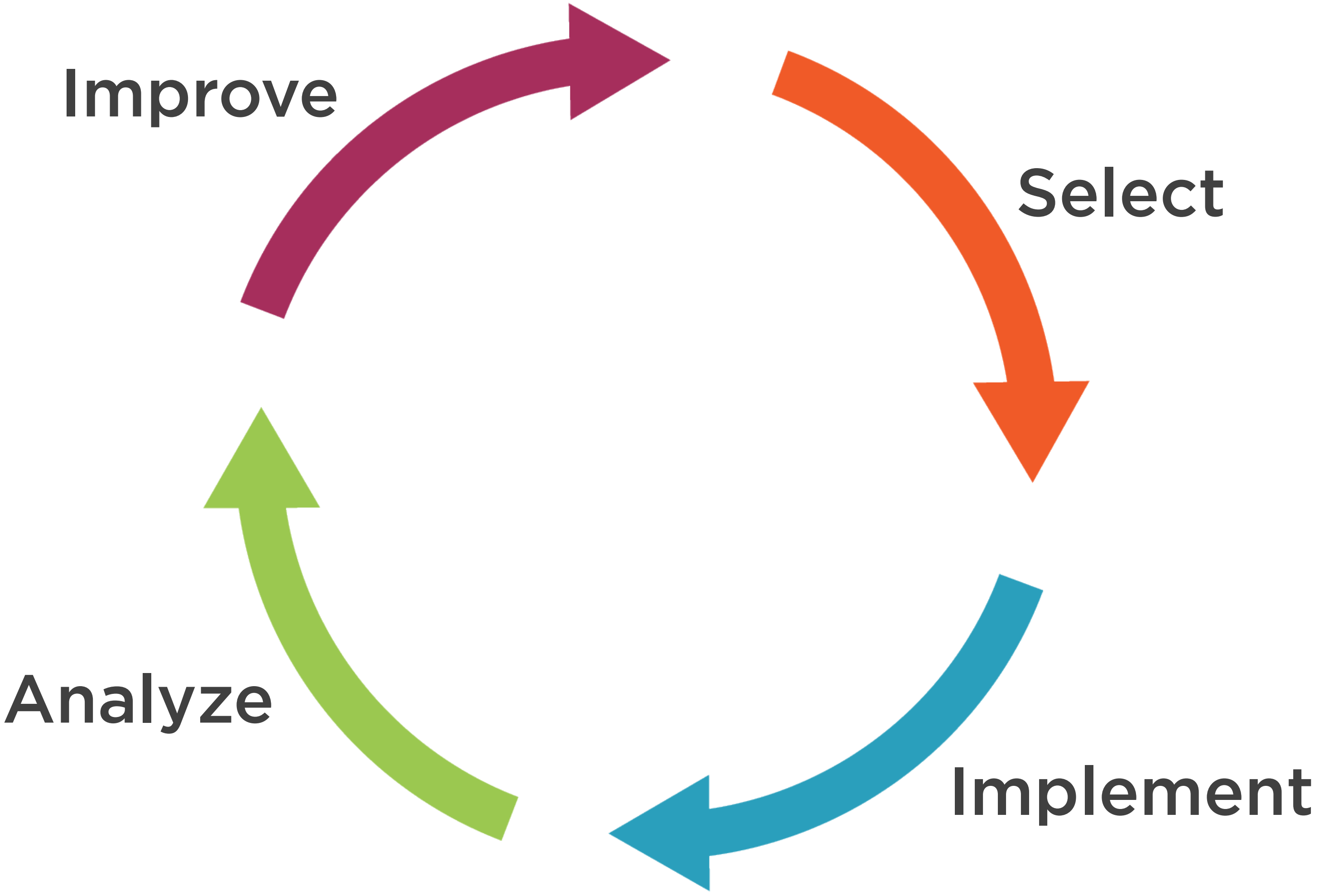
Compatibility

Depends on linter, programming language, and “quality” of linter

Trialability

Easy to employ in Continuous Integration pipelines

For All Tools: Iterative Process



More Information

https://github.com/hadolint/hadolint

https://github.com/PeterMosmans/tools-image/

Detecting Secrets

Why Detecting Secrets?

Should not be hardcoded

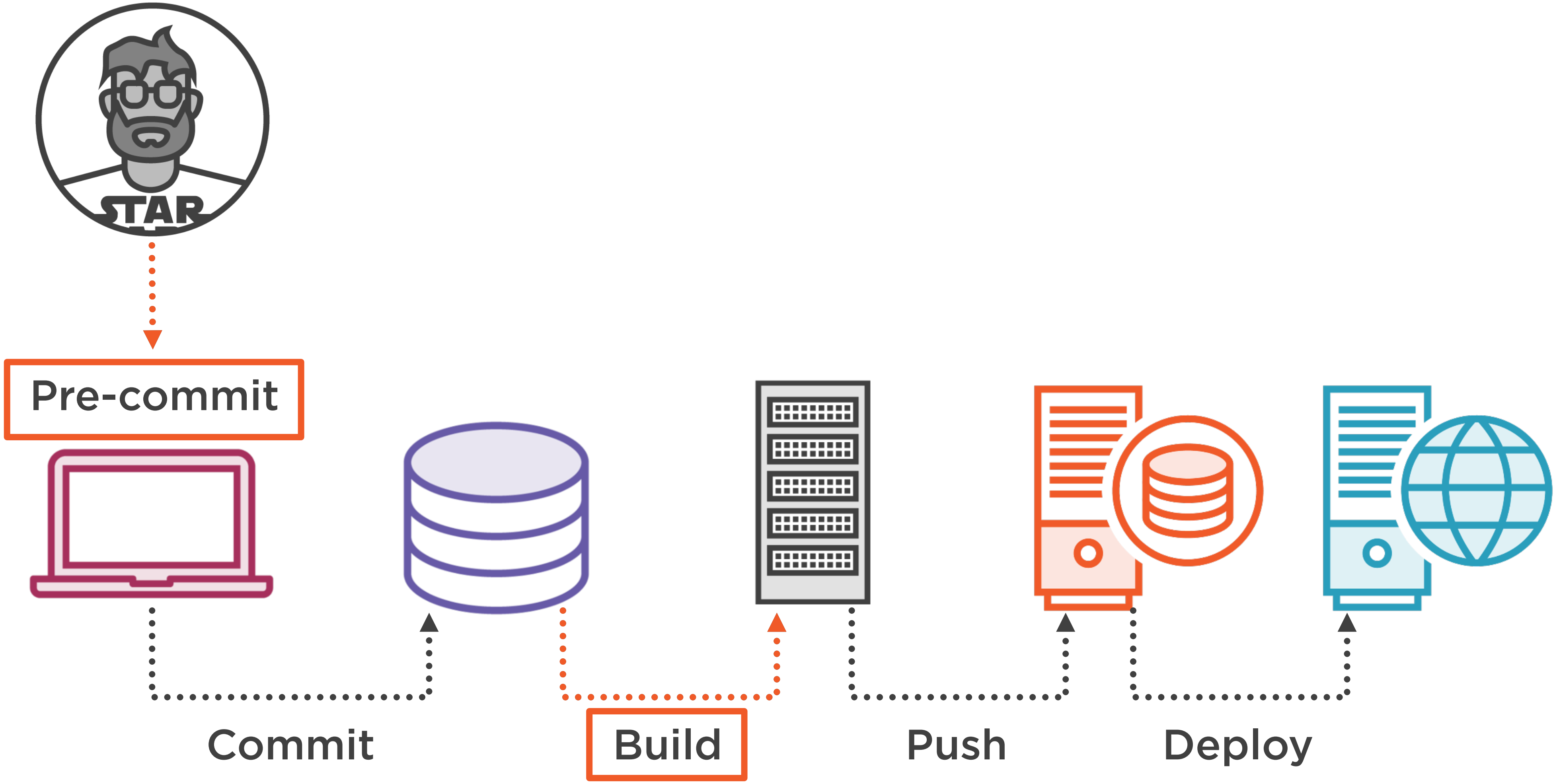
Should not be unencrypted

Should not be stored in source code

Or…

- Should be validated

Where and When to Detect Secrets



Tools That Will Be Demo-ed

truffleHog

- Searches through git repositories for secrets

pre-commit

- A framework to manage pre-commit git hooks

detect-secrets

- Detects secrets with options for setting baselines

Demo

Detecting existing secrets

- Install truffleHog
- Run truffleHog on tools-image
- Clone juice-shop project
- Run truffleHog on juice-shop

Demo

Detecting new and existing secrets

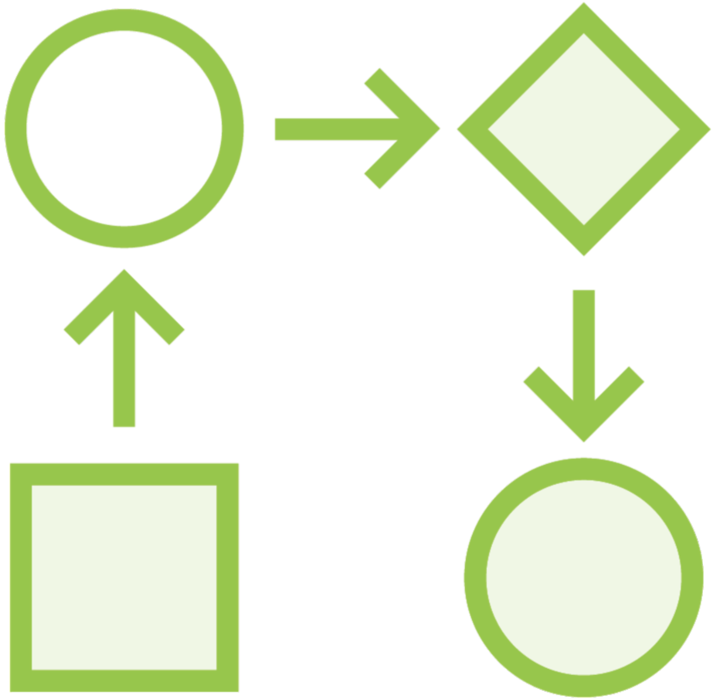
- Install and run detect-secrets
- Install pre-commit
- Configure pre-commit

Demo

Detecting new secrets during automated security testing

- Set up pipeline for juice-shop
- Use detect-secrets in the Jenkins CI/CD pipeline

Workflow for Detecting Secrets



First generate a baseline:

- Create list of current secrets
- Audit the list of secrets
  - Is it a false positive or a secret?

Add baseline to repository

Compare every scan with the baseline

- Warn or fail build when detecting new secrets

Update baseline when necessary

Detecting Secrets

Detects secrets in repositories

Pluggable architecture

Highly customizable

Advantage

Quick wins

Creates an overview of the current status

Makes it easy to gradually rollover

Compatibility

Most tools understand git

Works with plaintext files

Trialability

Easy to employ in Continuous Integration pipelines

Not much prerequisites

More Information

https://github.com/dxa4481/truffleHog

https://pre-commit.com/

https://github.com/Yelp/detect-secrets

https://github.com/bkimminich/juice-shop

Using Code Quality Systems

Wat Can A Code Quality Metrics System Do?

- Detect formatting or styling issues
- Suggest best practices
- Gives an “objectified” view of the state of the code
  - ..as well as over time
- Makes quality of code visible
- Increases overall quality of the code
- Makes maintenance of code easier

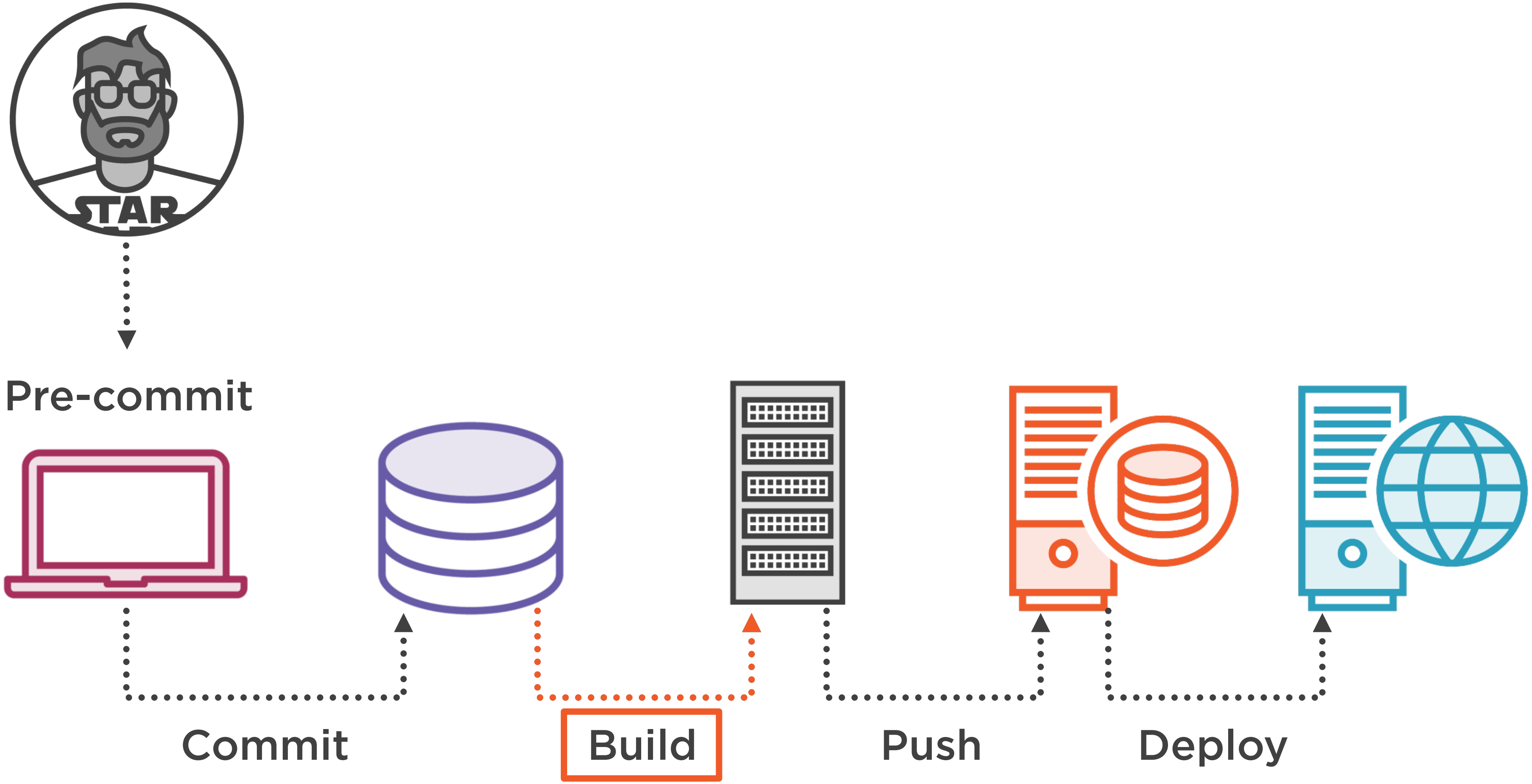
Issues With Code Quality Metrics Systems

Often resource-intensive and slow

Information overload can lead to focusing on “unimportant” issues

Metrics can give a false sense of (in)security

Where and When to Use Code Quality Systems

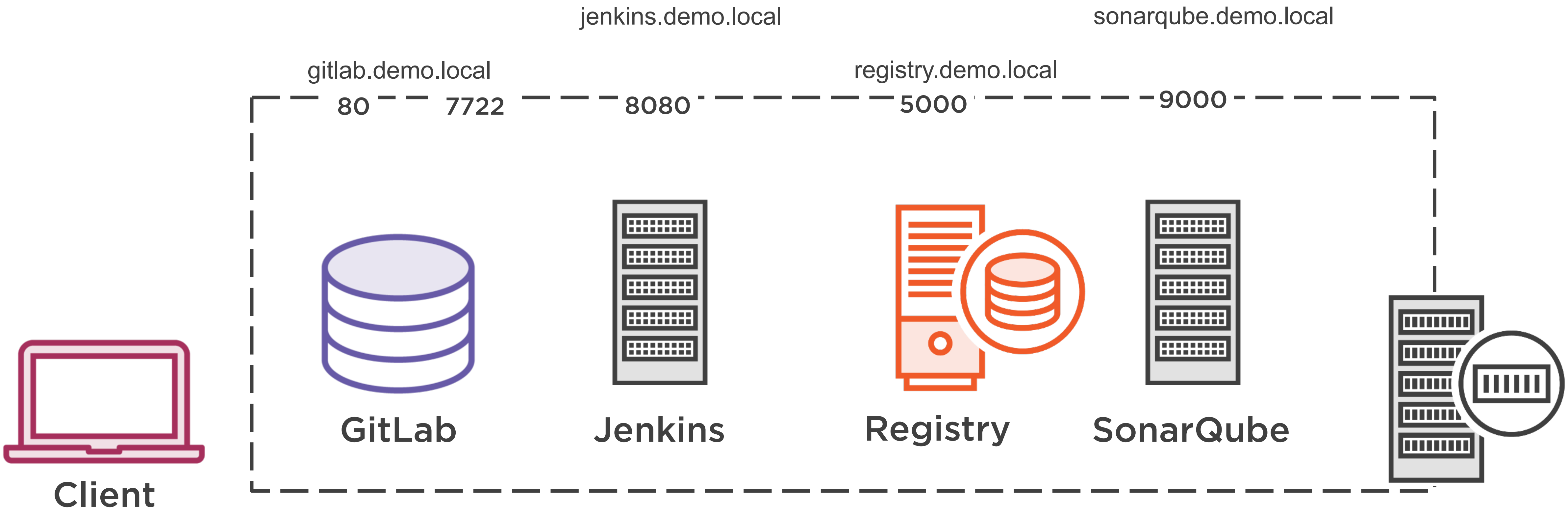


Tool That Will Be Demo-ed

SonarQube

- Code quality metrics tool

Demo Lab


jenkins.demo.local
sonarqube.demo.local
gitlab.demo.local
registry.demo.local
80
7722
8080
5000
9000
GitLab
Jenkins
Registry
SonarQube
Client

Demo

Installing a code quality metrics system:

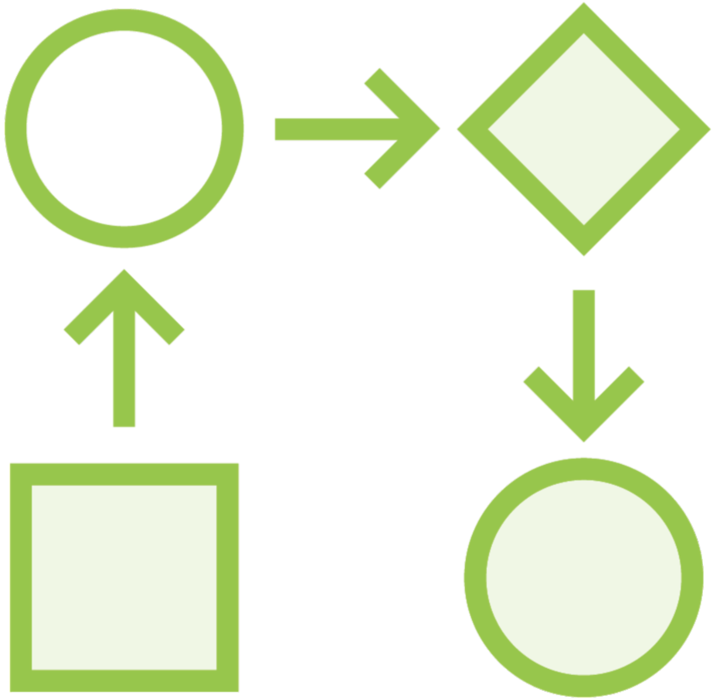
- Run and configure SonarQube
- Configure Jenkins

Demo

Using a code quality metrics system:

- Use SonarQube in a Jenkins CI/CD pipeline

Workflow for Code Quality Metrics Systems



Select a system with support for application's language and frameworks

Let system generate a list of current issues

Audit the list of issues

- Is it a false positive or an issue?
- Should it be shown?

Configure rules

Compare every scan with previous results

Code Quality Metrics System

Makes quality of code visible

Gives an objective view of the “state” of the code

Suggest best practices

Advantage

Graphical dashboard on code quality

Gives insight into impact of changes

Compatibility

Depends on language

Trialability

Set up moderately easy

Configuring and interpreting results is time-consuming

More Information

https://www.sonarqube.org

https://github.com/PeterMosmans/devsecops-lab/

Do not underestimate the time it takes to properly configure security testing tools

Summary

Linting

- Can give quick feedback
- Use strict versioning for linters

Detecting secrets

- Quick wins: Easy to implement

Code quality metrics systems

- Advanced reporting metrics
- Time-consuming to configure and use

Next Up

Maeve

“Some tools were easier to use than expected!”

“Great to hear that”

“Are you also interested in automating third party libraries security testing?”

“Absolutely, let’s go!”

Jennifer