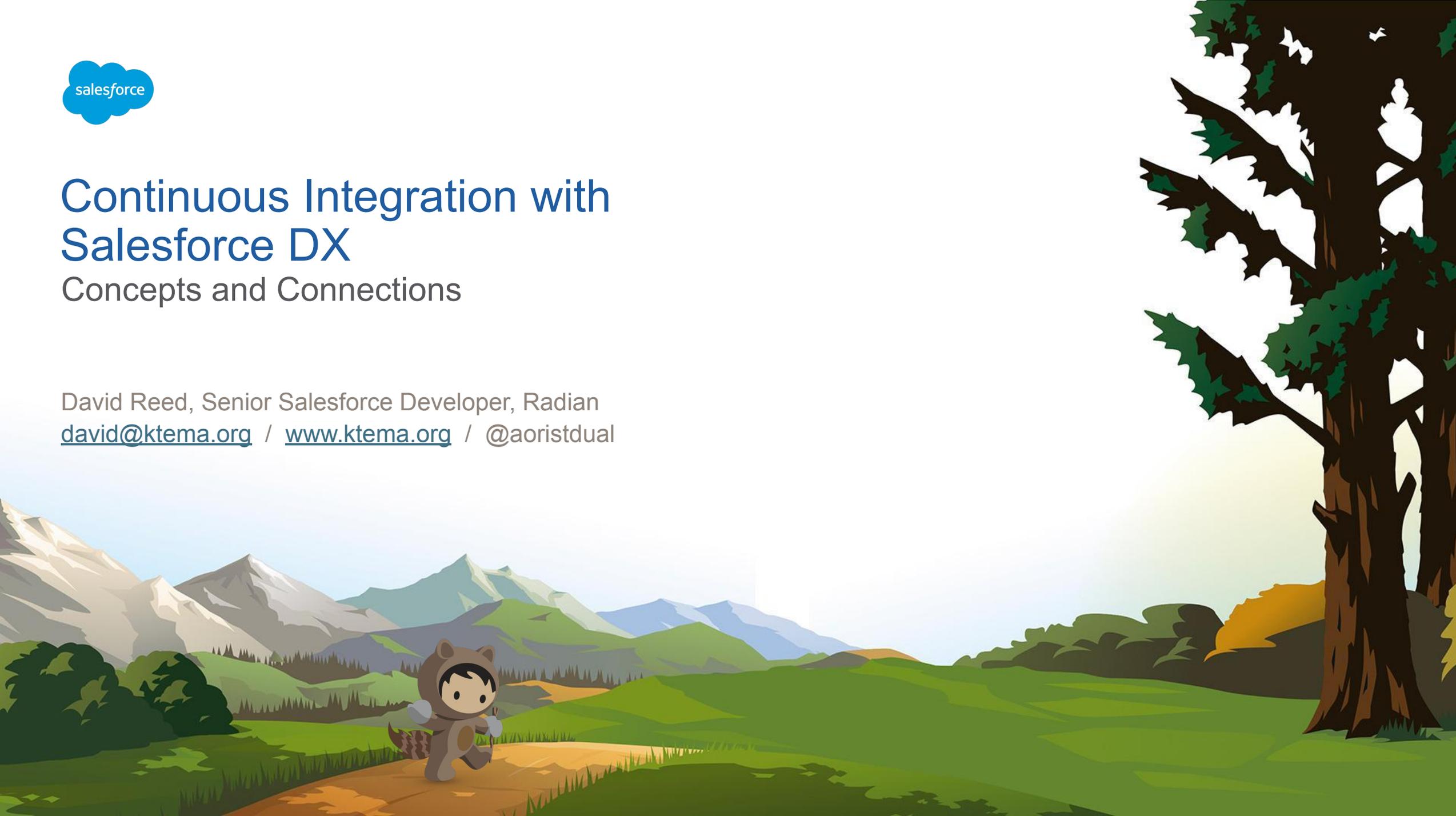


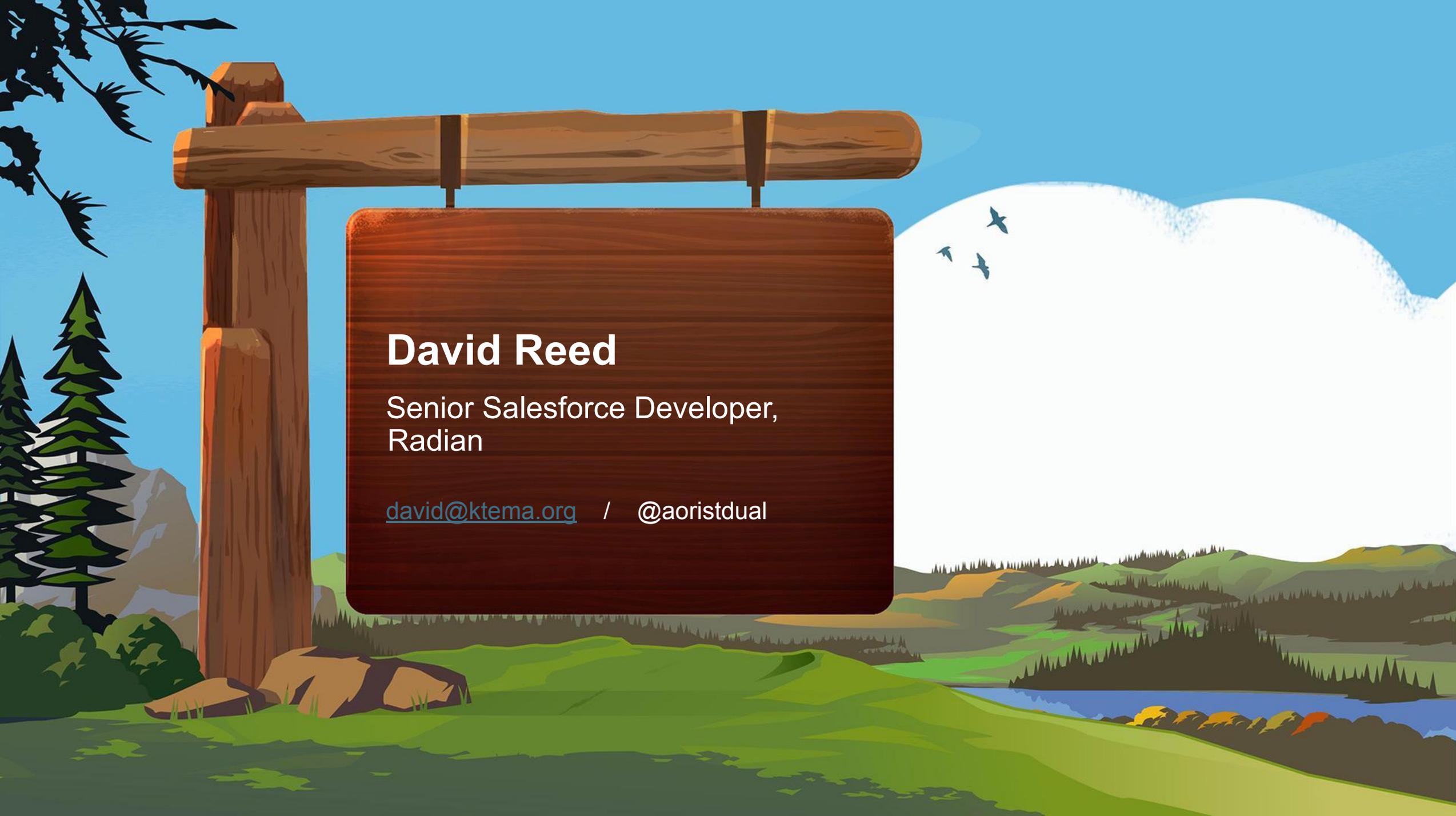


# Continuous Integration with Salesforce DX

## Concepts and Connections

David Reed, Senior Salesforce Developer, Radian  
[david@ktema.org](mailto:david@ktema.org) / [www.ktema.org](http://www.ktema.org) / @aorisdual



A wooden signpost stands in a scenic landscape. The signpost is made of two vertical wooden posts and a horizontal crossbar, with a dark brown rectangular sign hanging from it. The background features rolling green hills, a blue lake, and distant mountains under a bright blue sky with a large white cloud and three birds flying. On the left, there are several green evergreen trees and a large rock.

**David Reed**

Senior Salesforce Developer,  
Radian

[david@ktema.org](mailto:david@ktema.org) / @aoristdual

# Conceptual Introduction to Continuous Integration



# What Is Continuous Integration?

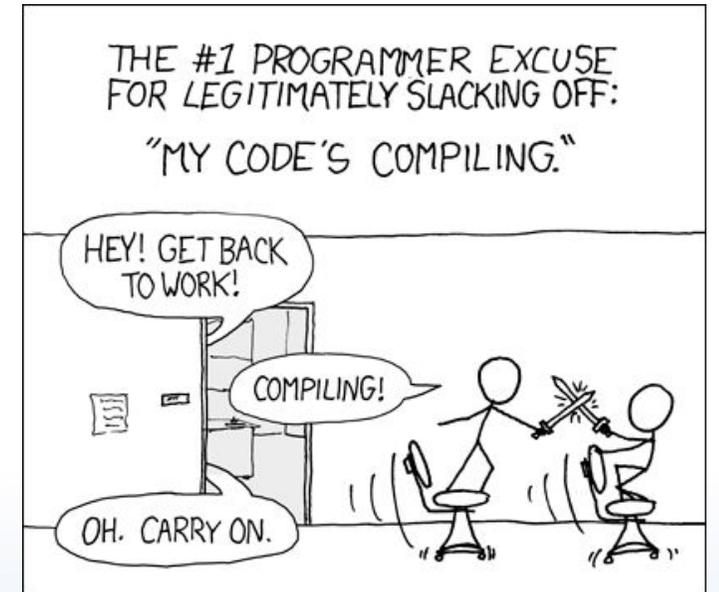


Methodology for building and testing software **constantly** and **automatically**

Use scripting to reduce or eliminate human intervention

Focus on **consistency, repeatability**

Add tools to increase insight and value



Your code is never **not** compiling.

# Continuous integration is for...



Developers who write code

Admins and app builders who work in an org that has code

Implementors and solution architects in code-based or code/declarative orgs

Release and application lifecycle managers

Leadership overseeing development activities

\* CI includes metadata too -- not just Apex!



# CI Flow: User Perspective



Write Code → Push → Get Feedback

```
$ git push
```



Test Summary Queue (00:00)

Your build ran **27** tests in unknown with **0 failures**  
**Slowest test:** MyProfilePageControllerTest testSave (took 0.22 seconds).

---

Show containers: All (1) Successful (1) Failed (0)

0  
(00:59)



# CI Flow: User Perspective



Finding and fixing a regression in under ~~two~~ ten minutes with CI

```
1 @isTest
2 public class AccountTriggerHandlerTest {
3     @isTest
4     public static void testAccountTriggerHandler() {
5         List<Account> testAccounts = new List<Account>();
6
7         for (Integer i = 0; i < 20; i++) {
8             testAccounts.add(new Account(Name = 'Dummy Account ' + String.valueOf(i), OwnerId = UserInfo.getUserId()));
9         }
10
11         insert testAccounts;
12
13         User u = [SELECT Name FROM User WHERE Id = :UserInfo.getUserId() LIMIT 1];
14
15         testAccounts = [SELECT Name FROM Account ORDER BY Name];
16
17         for (Account a : testAccounts) {
18             System.assert(a.Name.startsWith(u.Name + ': Dummy Account'));
19         }
20     }
21 }
```

# The Value Proposition

CI in the Development Lifecycle



# Visibility

Across your codebase, every commit

Tests passing? ✓

Regressions? ✓

Overall test coverage? ✓

Coding best practices? ✓

## All checks have passed

4 successful checks

✓		<b>ci/circleci: build</b> — Your tests passed on CircleCI!	<a href="#">Details</a>
✓		<b>ci/circleci: static-analysis</b> — Your tests passed on ...	<a href="#">Details</a>
✓		<b>codecov/patch</b> — Coverage not affected when com...	<a href="#">Details</a>
✓		<b>codecov/project</b> — 100% remains the same compa...	<a href="#">Details</a>



# Risk Reduction

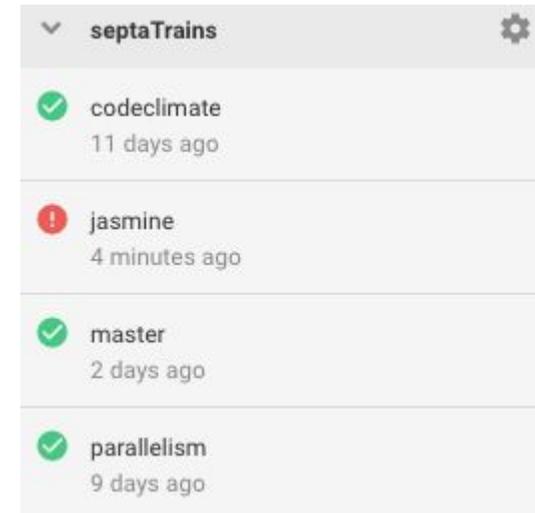


Never unknowingly merge broken code.

Find more regressions faster - no waiting on manual test cycles.

Surface areas of risk.

Iterate and branch without chaos.



A feature branch is failing, but master is green.



# Virtuous Feedback Cycle



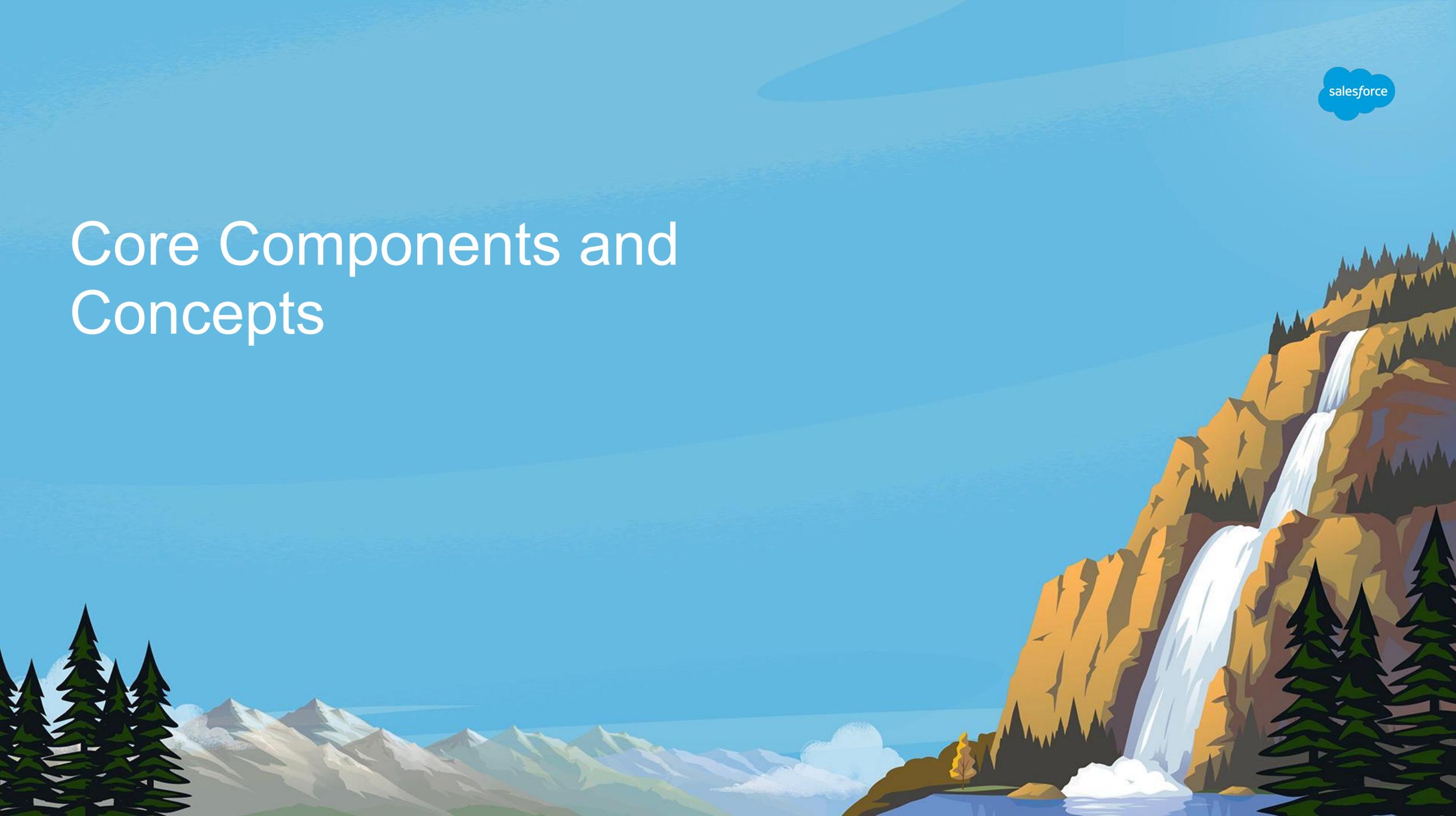
Utility of CI goes up with good development practices.

Value of good practices increases and becomes visible with CI.





# Core Components and Concepts



# CI Flow: User Perspective



```
$ git push
```

This is where we're digging in

Test Summary Queue (00:00)

Your build ran **27** tests in unknown with **0 failures**  
**Slowest test:** MyProfilePageControllerTest testSave (took 0.22 seconds).

---

Show containers: All (1) Successful (1) Failed (0)

0  
(00:59)



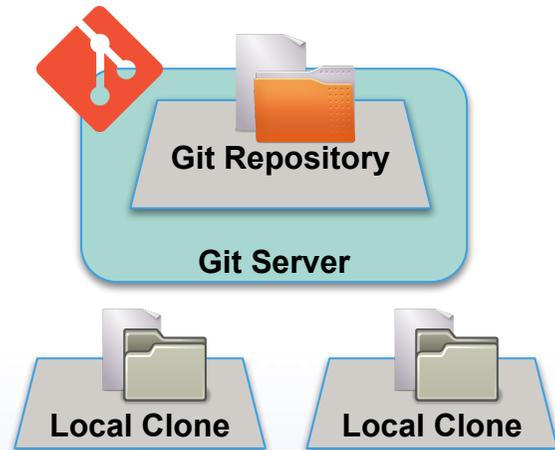
# Core Components of CI with Salesforce DX



## Version control

Server houses master repo, triggers CI process

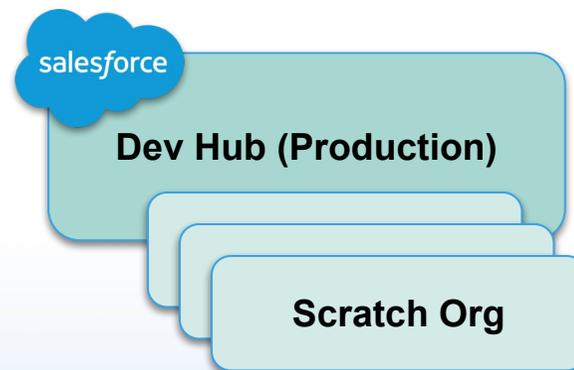
Developer repos hold working copy, connect to scratch orgs



## Salesforce DX

Production is the Dev Hub (controls scratch orgs)

Scratch orgs (~ Salesforce containers) for development and testing

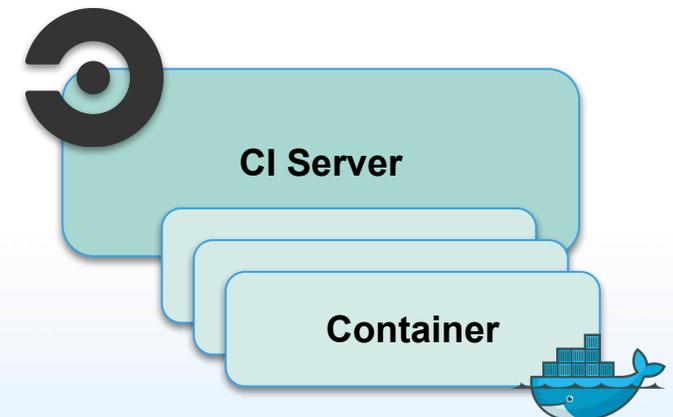


## CI provider

Watches Git server; runs flow on push

Executes steps defined by script

Build runs in virtualized environment, uses scratch org

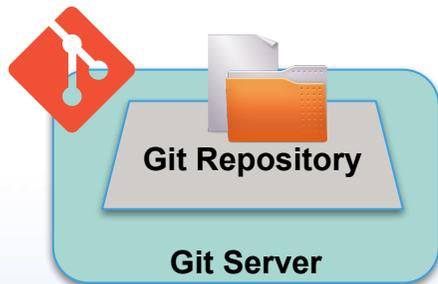
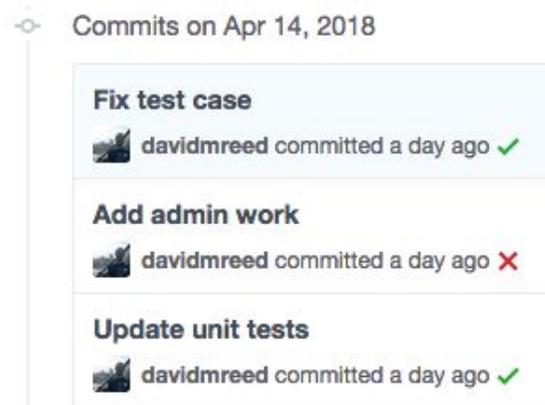


# Version Control: Source of Truth



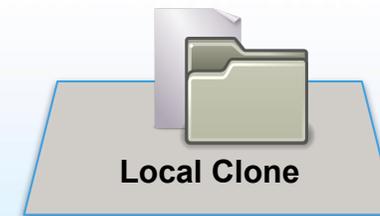
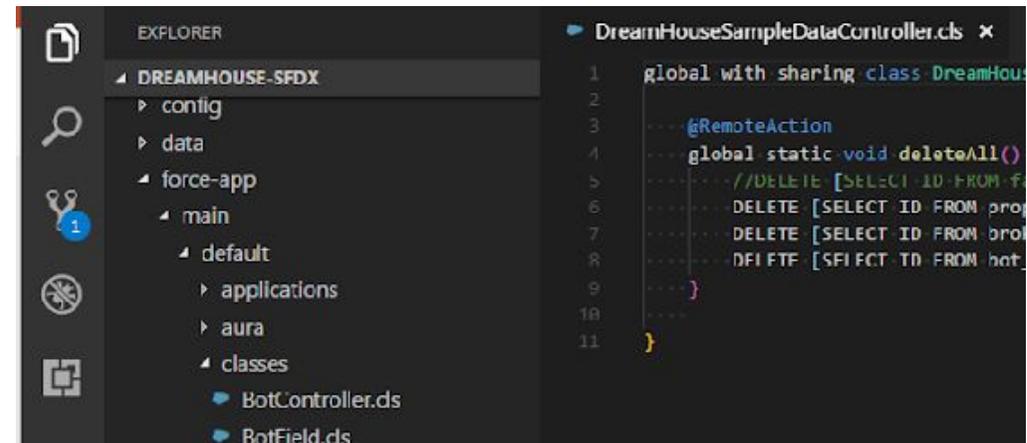
master lives on your version control server.

This is the team's source of truth.

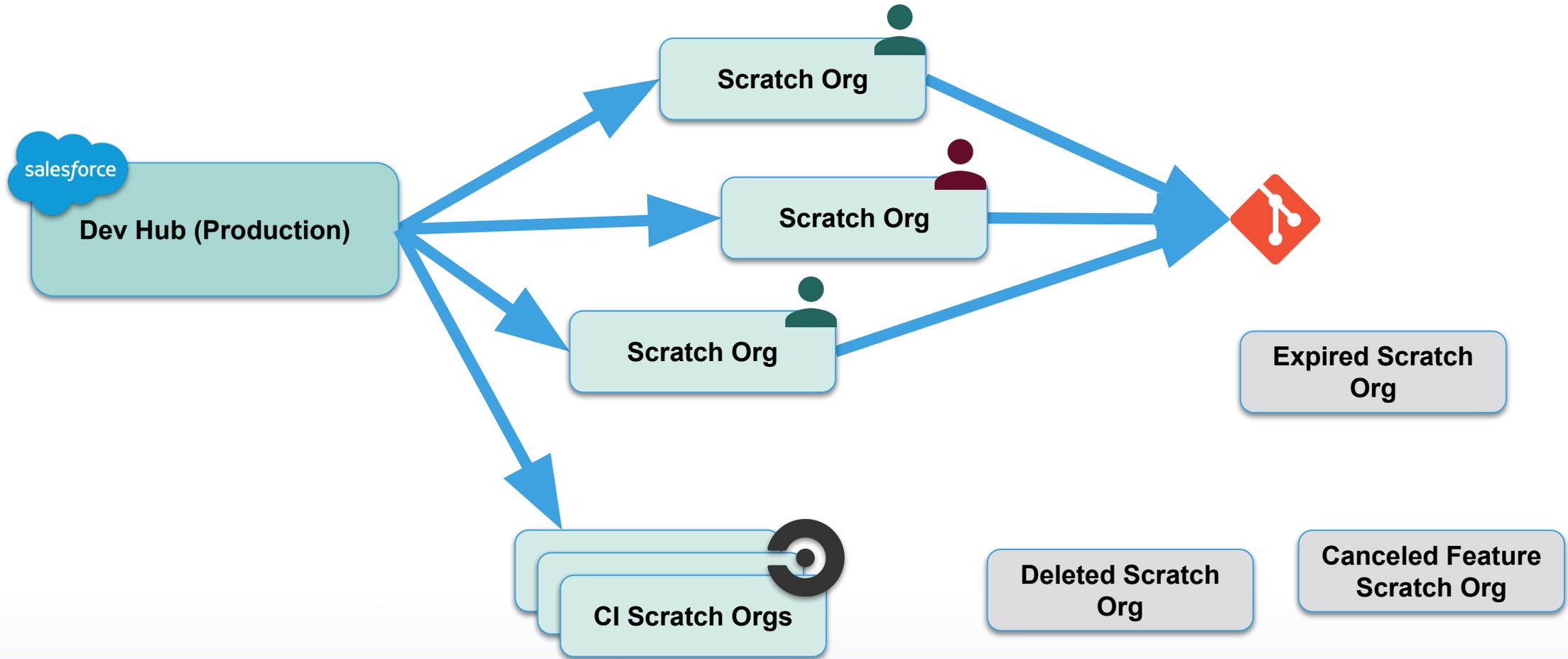


Your current work is built in an SFDX scratch org and tracked in your local Git repo.

These changes don't start CI until they're committed to Git and pushed to the server.



# SFDX Scratch Orgs: Location of Work and Testing



# Continuous Integration Provider: The Conductor



CI server reacts to changes in Git

Scope is configurable – run only on master, specific branches, etc.

SFDX scratch orgs take over some work done inside a traditional CI solution

CI orchestrates the build/test process



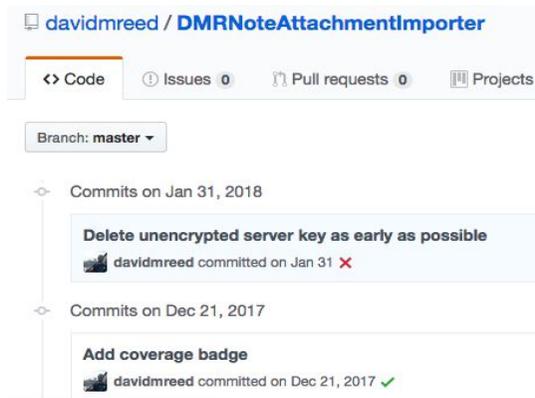
# Results & Visibility

Current status pushed everywhere you work



A screenshot of a CI build status card. At the top left, there is a green pill with a checkmark and the word "FIXED". To the right, the status is "Finished: 2 months ago (02:06)". Further right, it shows "Previous: 15", "Parallelism: 1x out of 4x", and "Queued: 00:00 waiting + 00:02 in queue". On the far right, it says "Resources: 2CPU". Below this, there is a section titled "COMMITTS (1)" with a commit by "David Reed" on "GitHub" with hash "fddce46" and the message "Delete unencrypted server key as early as possible".

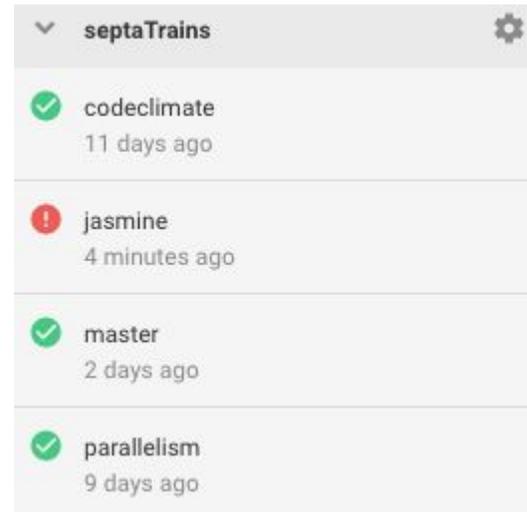
Build-level results in CI



A screenshot of a GitHub repository page for "davidreed / DMRNoteAttachmentImporter". The page shows the "Code" tab selected. Below the repository name, there are links for "Issues 0", "Pull requests 0", and "Projects". The current branch is "master". The commit history shows two commits: one on Jan 31, 2018, titled "Delete unencrypted server key as early as possible" by davidreed, and another on Dec 21, 2017, titled "Add coverage badge" by davidreed.

Branch-level status in CI

Commit-level status in VCS



A screenshot of a CI dashboard for a project named "septaTrains". The dashboard lists several build configurations with their status and last run time: "codeclimate" (passed, 11 days ago), "jasmine" (failed, 4 minutes ago), "master" (passed, 2 days ago), and "parallelism" (passed, 9 days ago). Each item has a green checkmark for success or a red exclamation mark for failure.

# Digging Deeper

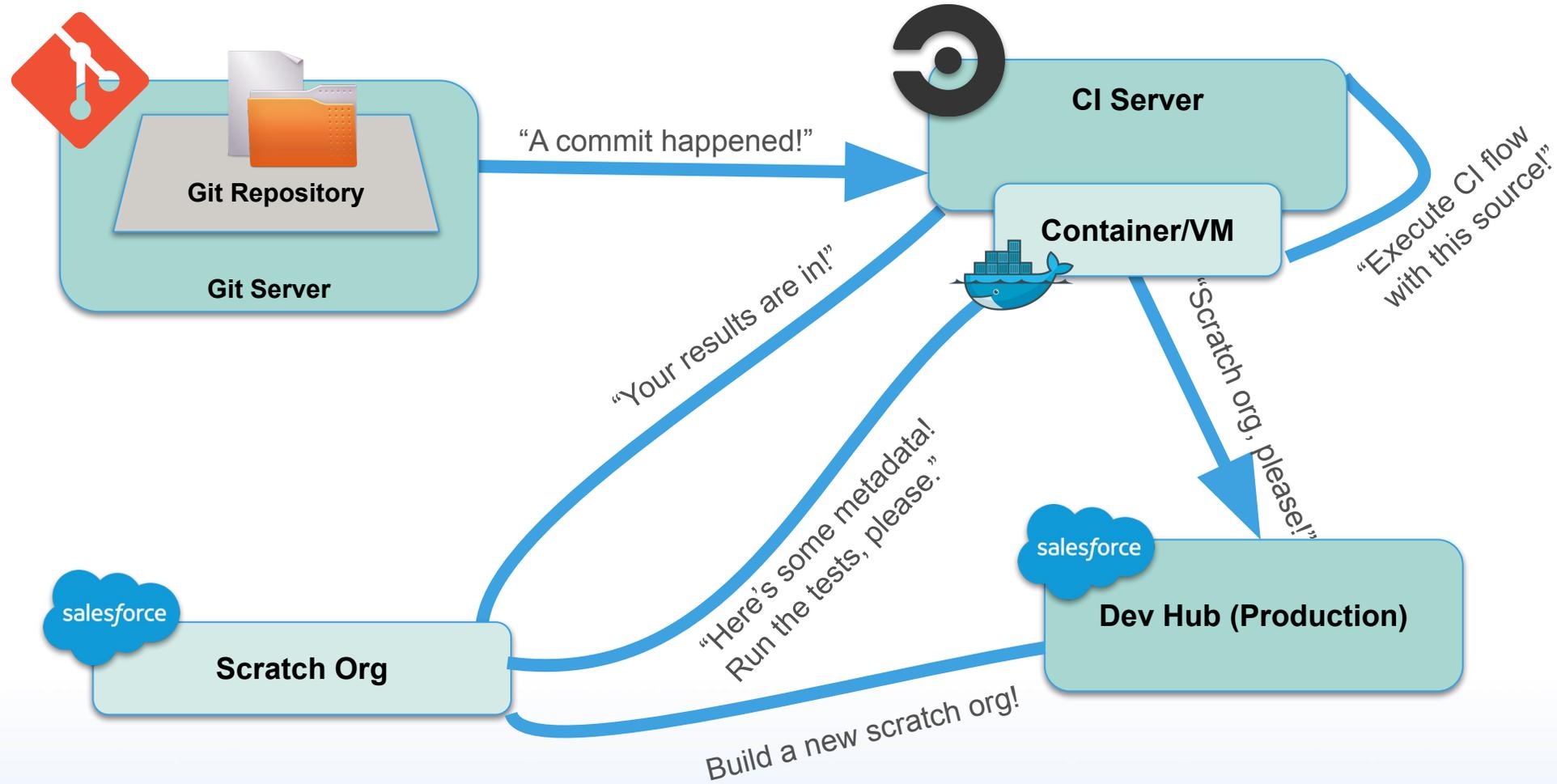
A Systems Perspective

The Salesforce logo, consisting of the word "salesforce" in white lowercase letters inside a blue cloud-like shape.

salesforce



# CI Flow: Connections Between Systems



# Containers, Scripts, and Connections

SFDX and CI rely on containers in different ways

CircleCI runs Bash scripts in Docker virtual environment

SFDX scratch orgs can be thought of like containers for Salesforce

Containerization and scripting allows plugging of toolchain



`%YAML 1.2`

`---`

`YAML: YAML Ain't Markup Language`





# What Defines an Org?

Broad categories, some overlap

## Edition and Feature Set

Developer / Enterprise / etc

Optional features like Person Accounts

Things you license from Salesforce – plus some aspects of Setup configuration

**Defined in JSON scratch org configuration**

## Configuration (Setup)

Changes you make in Setup that alter the behavior of your org

Feature-level configuration

**Split between JSON scratch org configuration, Metadata API source, and leftovers**

## Source and Metadata

Apex code

Visualforce

Lightning

Process/Flow/Workflow

Custom objects

**Defined in source code (Apex, JS, XML ...)**

# Shaping the SFDX Scratch Org



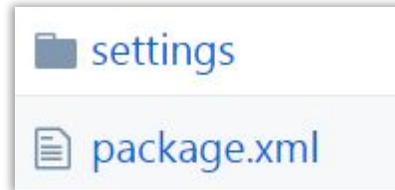
Start with JSON config file.

Deploy source to customize.

Edge-case tweaks with Metadata API or even Selenium.

```
{
  "orgName": "University of Caprica",
  "edition": "Enterprise",
  "features": ["Sites", "Communities", "PersonAccounts"],
  "description": "New architecture for defense mainframe",
  "hasSampleData": "false",
  "orgPreferences" : {
    "enabled": ["NetworksEnabled", "ChatterEnabled"]
  }
}
```

config/project-scratch-def.json



```
- run:
  name: Perform Metadata Deploy
  command: |
    sfdx force:mdapi:deploy -d md-src -w 5
```

Metadata API Deployment



# What Defines a Container?

Broad categories, more overlap

## Operating System

Typically Ubuntu or another Linux distribution

**Defined by CI setup and/or image selection**

## Software

Available packages, like SFDX, Java, node.js

**Defined by image selection, scripted in build steps**

## Source and Metadata

Apex code

Visualforce

Lightning

Process/Flow/Workflow

Custom objects

**Checked out from Git**



# Shaping the CI Container

Pick base image (Java, NPM, etc.)

Customize with `apt-get`, `npm install`, other scripts

Reproducible setup - reproducible results

```
docker:  
  - image: circleci/python:3.6.4-node
```

Selecting Docker image in `config.yml`

```
- run:  
  name: Install Salesforce DX and simple_salesforce  
  command: |  
    if [ ! -d node_modules/sfdx-cli ]; then  
      export SFDX_AUTOUPDATE_DISABLE=true  
      export SFDX_USE_GENERIC_UNIX_KEYCHAIN=true  
      export SFDX_DOMAIN_RETRY=300  
      npm install sfdx-cli
```

Using `npm` to install `sfdx`



# Scripting Build Steps

(Almost) any tool, any command, any connection

Build steps can:

- Check out source code
- Install software in container
- Connect to SFDX scratch org
- Load data
- Run unit tests
- Run off-platform code
- More to come later

```
- run:
- run:
  name: Integration Test - Access Token
- run:
  name: Load Data
  command: |
    sfdx force:data:tree:import -p TreePlan.json -u scratch
    sfdx force:data:bulk:upsert -s Account -f ./assets/account.csv
    export INSTANCE_URL=$(node_modules/s
    python example-simple-salesforce.py
```

# Key Principles

A bespoke CI process, from off-the-shelf tools

Scripting + containers = **automation** and **reproducibility**.

Once you're scripting with CI, you have enormous power to add tools.

Taking advantage of prior art, then innovating.



# New Tools

A Tasting Menu

The Salesforce logo, consisting of the word "salesforce" in a white, lowercase, sans-serif font, enclosed within a blue, stylized cloud shape. The background of the slide is a colorful illustration of a mountain landscape. A brown bear wearing a white tank top stands on a dark grey rock ledge in the foreground, looking up at a white goat climbing a rope on a yellow-orange cliff face. Above the goat, a person in a red hat and brown gear is also climbing the rope. The background features a blue sky with soft clouds and a blue body of water. A green tree is visible on the left side of the frame.

# Code Coverage Monitoring and Visualization



Codecov.io

The screenshot displays the Codecov.io interface for a Salesforce project. At the top, a commit by 'davidmreed' 3 months ago is shown with a 'CI Passed' status and a '100.00%' coverage badge. The breadcrumb path is '/ force-app / main / default / classes / DMRImportNotesPageController.cls'. The main editor shows the source code for 'DMRImportNotesPageController.cls' with line numbers 1 through 22. Each line is highlighted in green, indicating 100% coverage. The code includes a class definition with methods for 'batchSize', 'importType', 'removeImported', 'hasSelection', and 'hasRecords'. On the right, a 'Coverage' table summarizes the coverage for the file and the project.

Coverage	
DMRImportNotesPageController.cls	100.00%
Folder Totals (3)	100.00%
Project Totals (1)	100.00%

# Static Analysis with PMD



PMD static analysis can run in parallel with scratch org deploy and test.

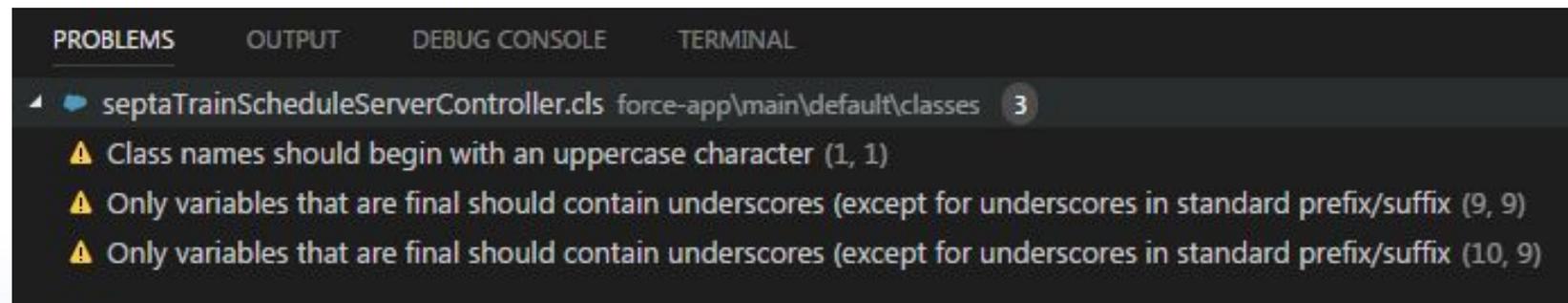
Block the build on SOQL/DML in loops, more.

PMD can run in your IDE or in the cloud.

## All checks have passed

4 successful checks

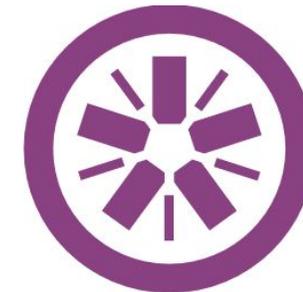
✓		ci/circleci: build — Your tests passed on CircleCI!	<a href="#">Details</a>
✓		ci/circleci: static-analysis — Your tests passed on ...	<a href="#">Details</a>
✓		codecov/patch — Coverage not affected when com...	<a href="#">Details</a>
✓		codecov/project — 100% remains the same compa...	<a href="#">Details</a>



# Lightning Testing Service

Unit test JavaScript with Mocha or Jasmine in SFDX scratch orgs

```
Jasmine 2.6.1  
.....  
6 specs, 0 failures  
  
septaTrainTrackerController  
  sets selected entity upon event receipt  
  updates the map view upon navigate  
  
septaStationScheduleController  
  sets schedule and display name upon selection  
  
septaScheduleSidebarController  
  sets selected entity upon event receipt  
  
septaEntitySelectorController  
  fires event upon navigate  
  sets select options from the controller
```



# Testing Multiple Org Types and Shapes

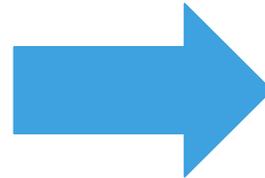


```
{} project-scratch-def.json •
1  {
2  |   "orgName": "David Reed",
3  |   "edition": "Enterprise",
4  |   "features": ["Sites"],
5  |   "orgPreferences" : {
6  |     "enabled": ["S1DesktopEnabled"]
7  |   }
8  }
9
```

Scratch definition JSON file(s)

```
workflows:
  version: 2
  test_and_static:
    jobs:
      - build-enterprise
      - build-developer
      - static-analysis
```

Workflow definition



master / test\_and\_static  
Move to SFDX 6.8.2  
Rerun

3 jobs in this workflow

- ✓ build-enterp... ⌚ 01:32
- ✓ static-analy... ⌚ 00:06
- ✓ build-develo... ⌚ 01:44

Results per org type in CircleCI and GitHub



# Testing Software that Integrates with Salesforce

Salesforce DX isn't limited to Apex code running on the Force.com platform.

Code that integrates to Salesforce can be tested against scratch orgs too.

```
- run:
  name: Integration Test - Username and Password
  command: |
    . venv/bin/activate
    node_modules/sfdx-cli/bin/run force:user:password:generate > /dev/null
    export PASSWORD=$(node_modules/sfdx-cli/bin/run force:user:display --json | pyth
    export SF_USERNAME=$(node_modules/sfdx-cli/bin/run force:user:display --json | p
    python example-simple-salesforce.py -p "$PASSWORD" -u "$SF_USERNAME" -t "" -s
```

Integration testing a Python/simple\_salesforce application

# Building Packages for Continuous Deployment



“Gate” build steps conditioned on successful tests or approval

Build in sophisticated branching and tagging strategy

Create new Unlocked Package versions for deployment

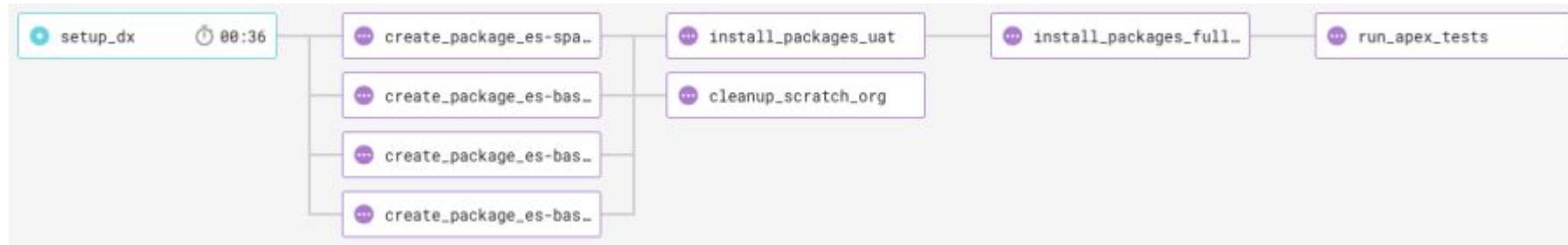


Image via Salesforce Developer Blog  
(<https://developer.salesforce.com/blogs/2018/06/working-with-modular-development-and-unlocked-packages-part-4.html>)

# Links and Resources



## **Getting Started with Salesforce DX (5-part series):**

`developer.salesforce.com/blogs/2018/02/getting-started-salesforce-dx-part-1-5.html`

## **Continuous Integration with Salesforce DX Trailhead:**

`trailhead.salesforce.com/en/modules/sfdx_travis_ci`

## **Templates on GitHub:** `github.com/davidmreed/circleci-sfdx-examples`

*(includes all code from this presentation)*

## **David Reed:**

[www.ktema.org](http://www.ktema.org)    [david@ktema.org](mailto:david@ktema.org)    [@aoristdual](https://twitter.com/aoristdual)



THANK YOU

