DevSecOps In Real Life

From Concept to Code

SAFe SUMMIT
October 27-28, 2020

CHAT WITH ME LIVE DURING SESSION

Marc Rix

SAFe Fellow, Framework Team

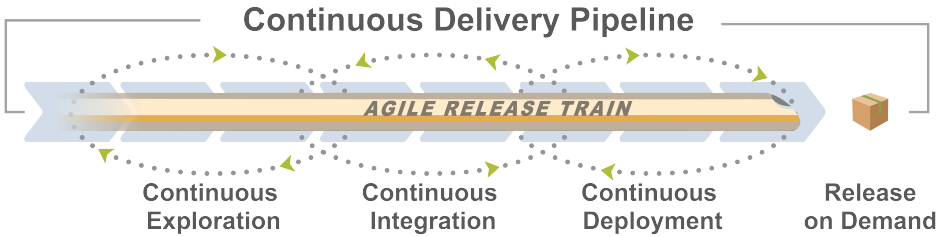
Scaled Agile, Inc.

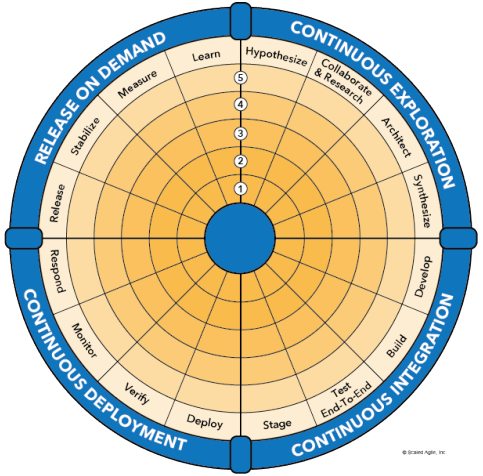
In partnership with addiction STUDIO

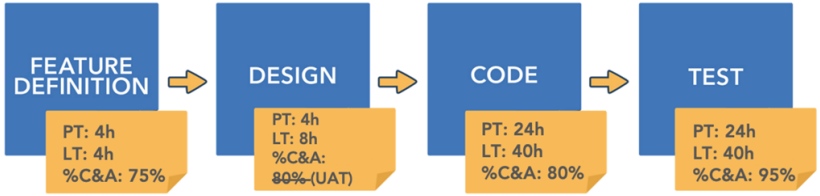
SCALED AGILE PARTNER GOLD

The Concepts

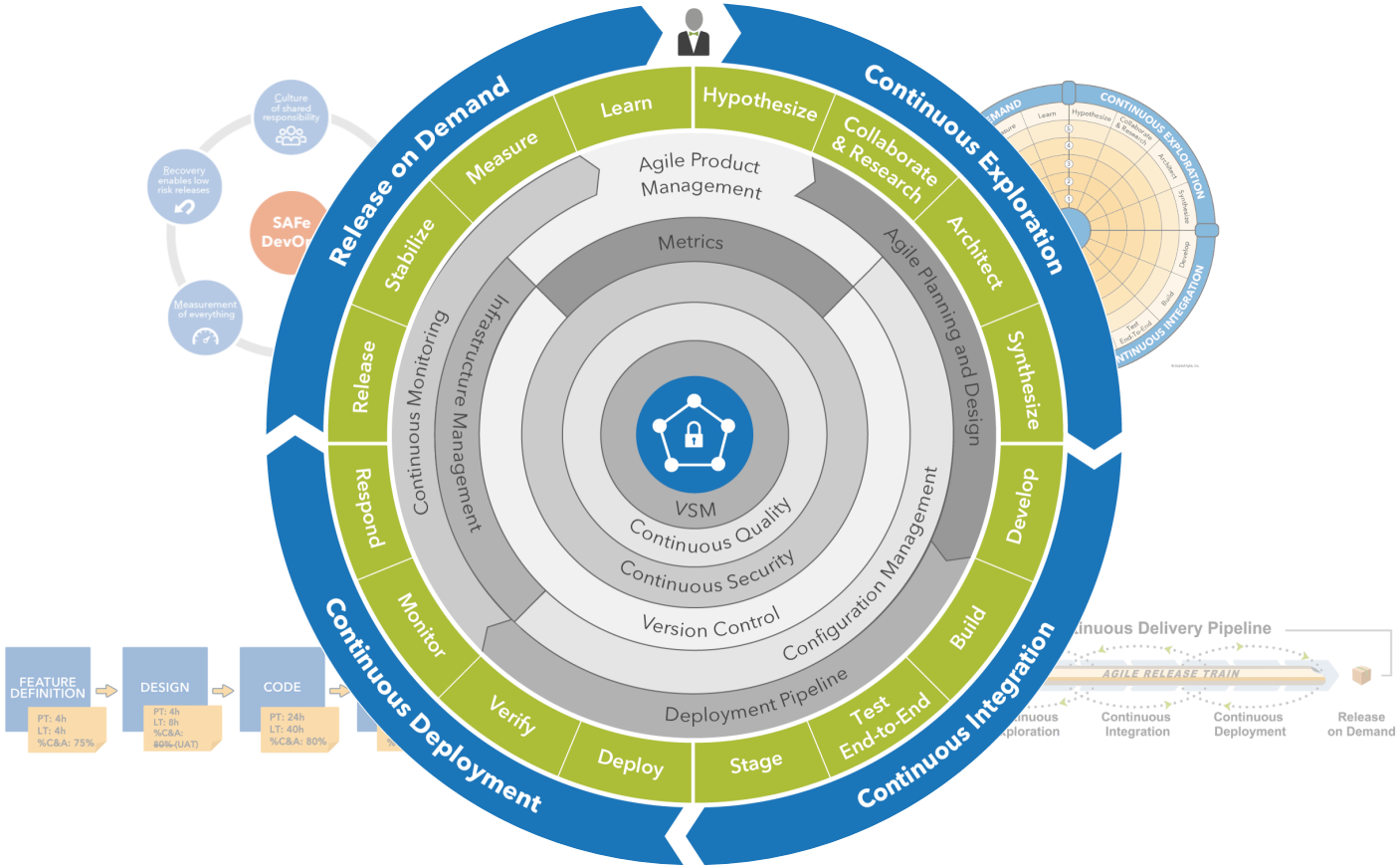
Current: Many standalone DevOps concepts



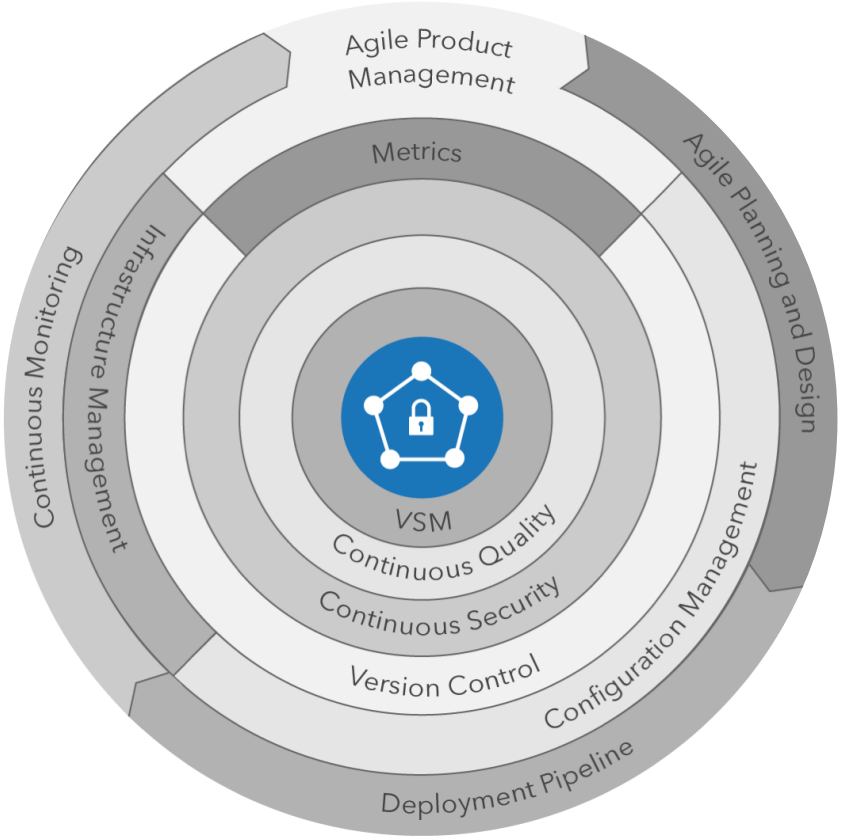




Coming soon: one unified DevSecOps concept



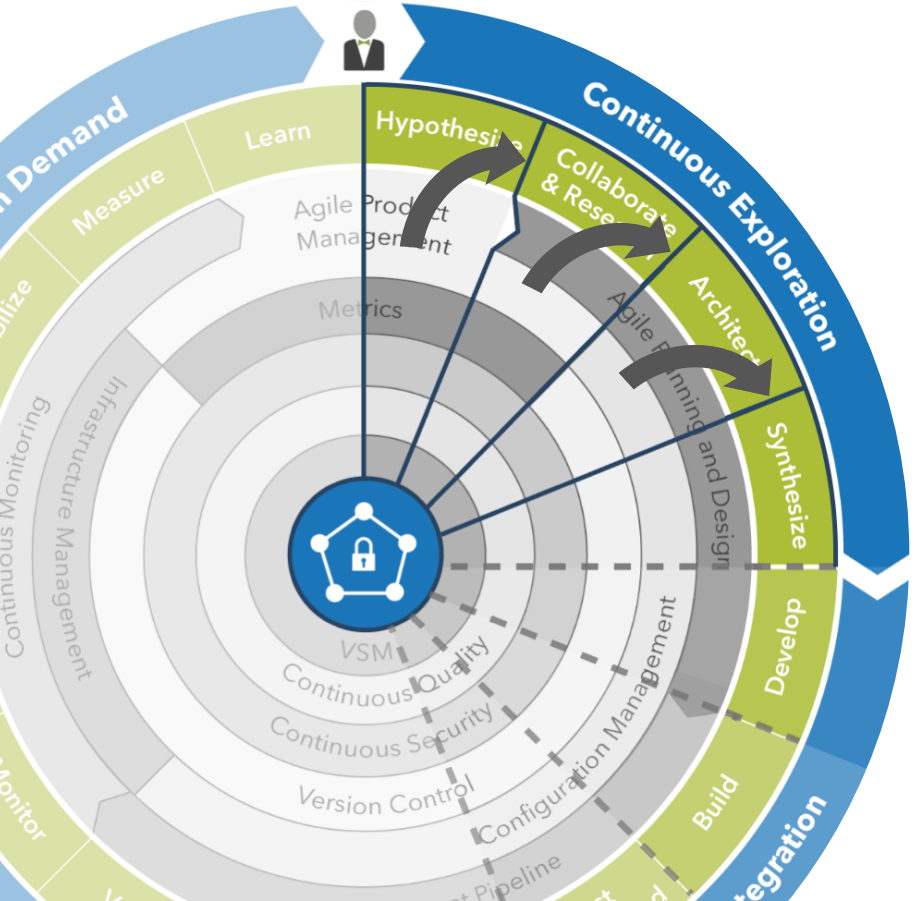
New: Technical practices!



12 essential DevSecOps "disciplines" that implement and enable the CDP

Underneath are the specific practices and tools that drive continuous delivery at scale.

How DevSecOps fuels continuous delivery

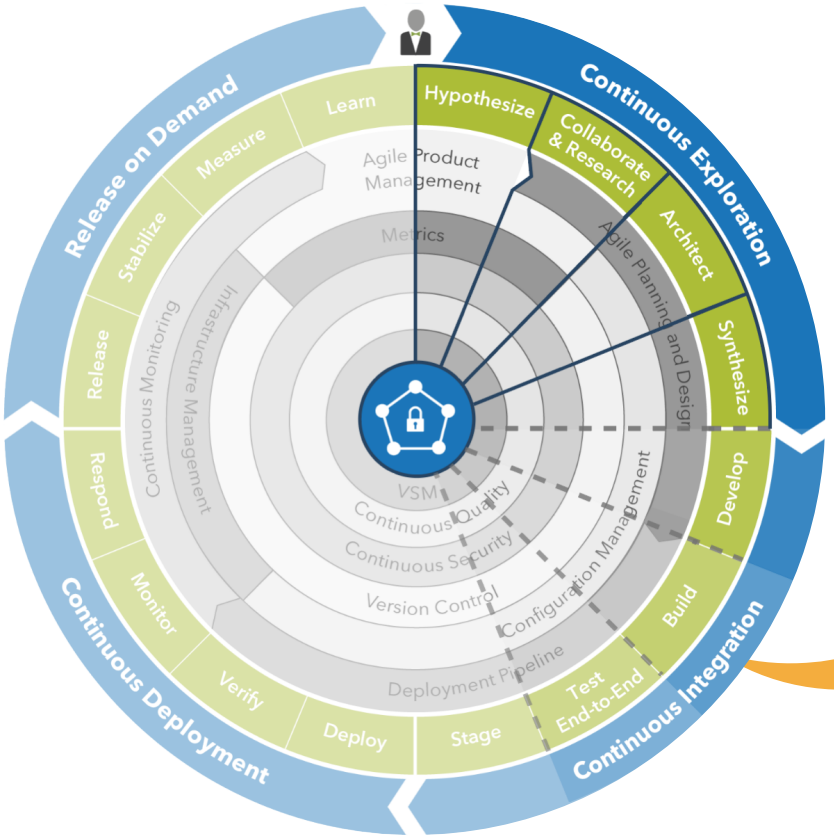


The CDP pulls in technical practices from multiple disciplines at each step.

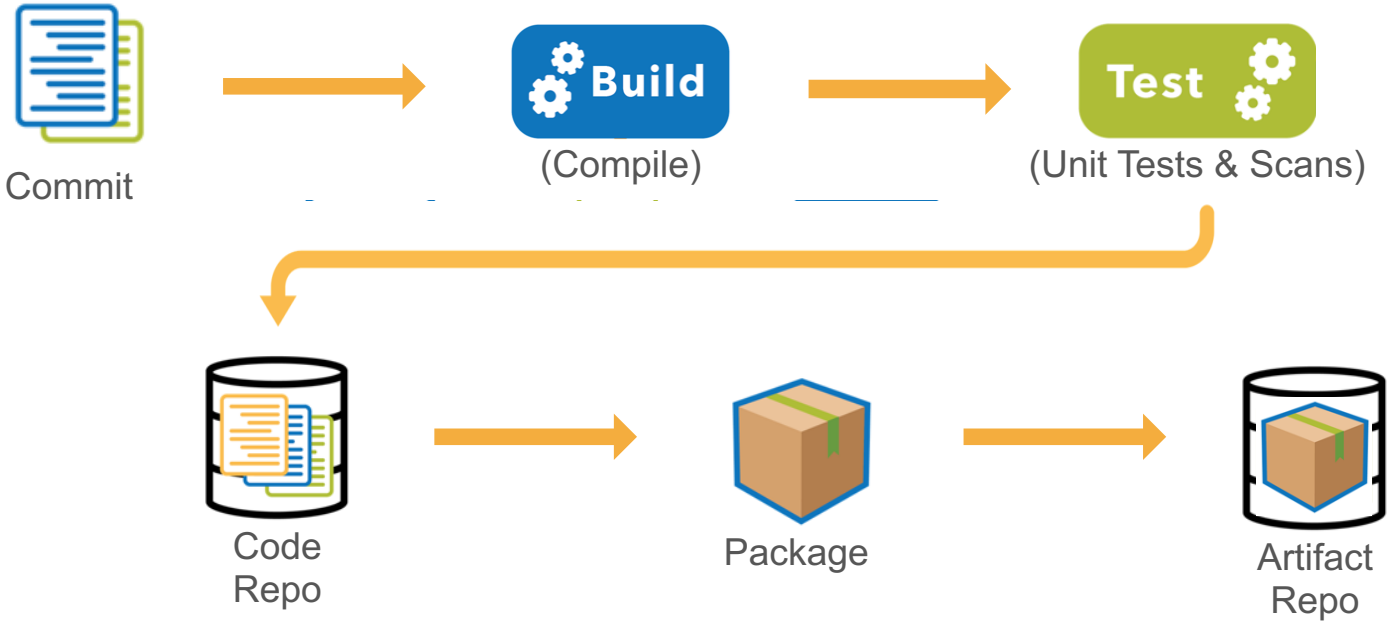
DevSecOps aligns those practices to drive continuous collaboration and flow throughout the CDP.

Example: The Build Step IRL

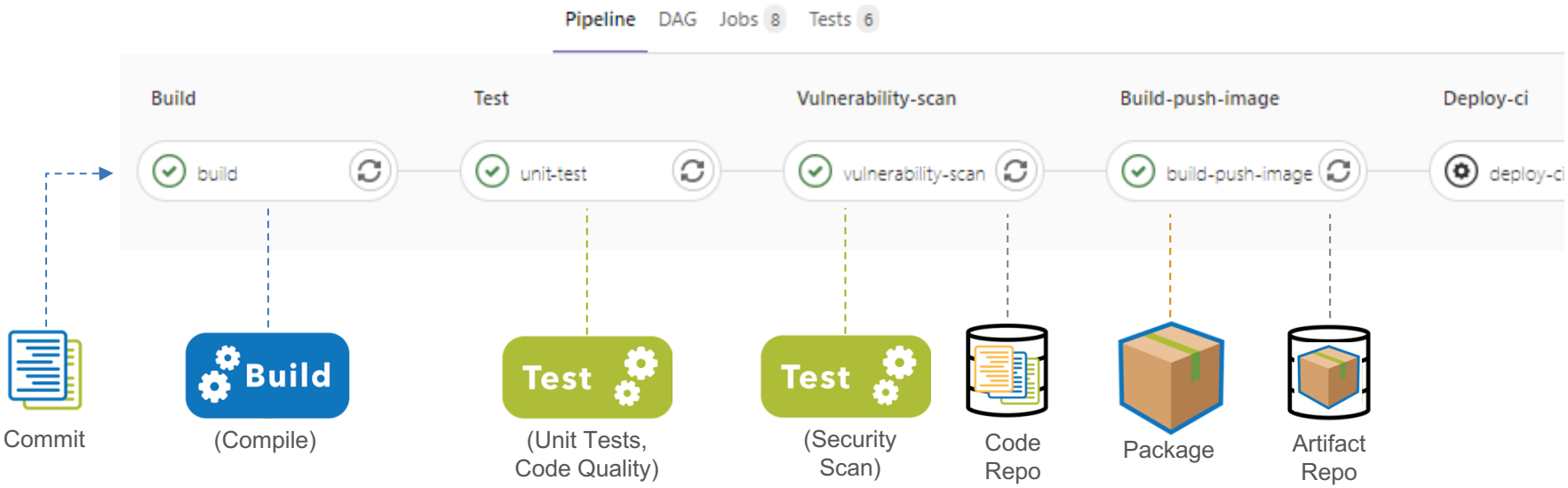
Build practices in real life



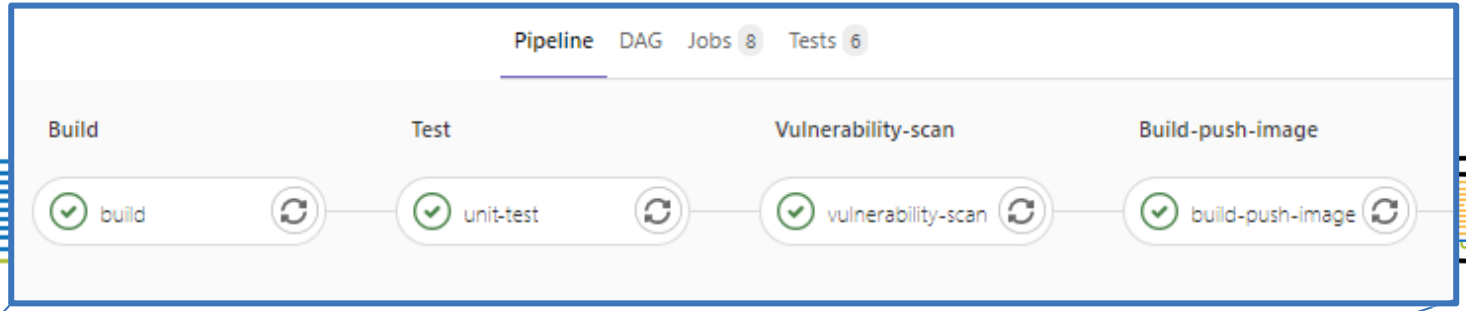
The Build step in SAFe



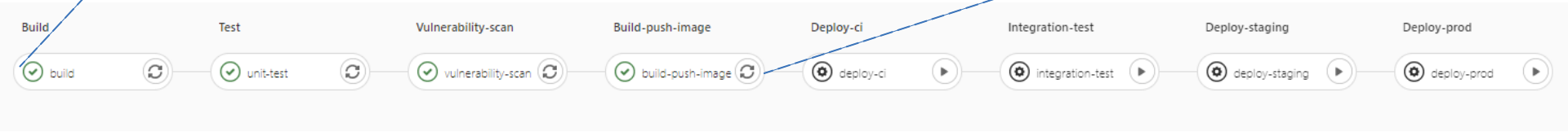
The Build step orchestrated in a CI system



Deployment Pipeline IRL



Build portion (from *commit* to *artifacts in repo*")



Full deployment pipeline (from *commit* to *successfully deployed to prod*)

Configuration Management IRL

.gitlab-ci.yml 1.94 KB

```yaml
variables:
  DOCKER_HOST: tcp://localhost:2375
  DOCKER_TLS_CERTDIR: ""
  GIT_CLEAN_FLAGS: none
  SONAR_LOGIN: admin
  SONAR_PASSWORD: admin
  SONAR_HOST_URL: "https://sonarqube-$TEAM_NAME.scaled-agile-

image: docker:19.03.0

services:
  - docker:dind

# cache:
#   paths:
#     - node_modules/

stages:
  - build
  - test
  - vulnerability-scan
  - build-push-image
  - deploy-ci
  - e2e-test
  - deploy-staging
  - deploy-prod
```

Pipeline Stages

```yaml
build:
  image: trion/ng-cli-karma
  stage: build
  script:
    - npm install
    - ./node_modules/@angular/cli/bin/ng build --progress false --prod
  artifacts:
    paths:
      - dist/*
```

Tools invoked in each stage

```yaml
vulnerability-scan:
  image: sonarsource/sonar-scanner-cli:latest
  stage: vulnerability-scan
  allow_failure: true
  script:
    - sonar-scanner -Dsonar.projectKey=company-x-feedback-ui -Dsonar.sources=src

build-push-image:
  stage: build-push-image
  needs: ["build",test, vulnerability-scan]
  script:
    - docker login -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD $CI_REGISTRY
    - docker build -t $CI_REGISTRY_IMAGE:latest .
    - docker push $CI_REGISTRY_IMAGE:latest
```

Version Control IRL

Name	Last commit	Last update
.vscode	Initial commit	2 hours ago
e2e	Initial c	2 hours ago
src	Initi	2 hours ago
.browserslistrc	Initial commit	2 hours ago
.editorconfig		
.gitignore		
.gitlab-ci.yml		
Dockerfile		
karma.conf.js		
nginx.conf	Initial commit	
package-lock.json		
package.json		
sonar-project.properties		
stage1.groovy	Initial commit	
stage2.groovy	Initial commit	

Source code

Pipeline config (IaC)

Test framework config

Web server config

Tool chain versions

Security scan config

Pipeline stage config

Container config

Continuous Security IRL

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

1	Bugs			Reliability	C
2	Vulnerabilities			Security	D
1	Security Hotspots	0.0%	Reviewed	Security Review	E
26min	Debt	5	Code Smells	Maintainability	A

Security and code quality scans execute on every commit.

Continuous Quality IRL (test definition)

src > test > resources > features > collect-feedback.feature

Feature acceptance criteria captured in file

```
Feature: collect feedback

    Allow a server to initiate a new feedback collection session for a customer.
    Next, present the customer with a series of questions allowing them to provide
    feedback in the form of numbers ranging from 1 to 10.

    Scenario: Initiate a new feedback collection session
        Given Table number "10" and server 1
        When requesting a new feedback collection session
        Then a new visit is created and returned

    Scenario: Retrieve the first question for a new visit
        Given a new visit
        When the next question is requested
        Then the first question is returned

    Scenario: Submitting an answer to a question
        Given a response of 7 to question 1 for visit 1
        When submitting the response
        Then the response is stored and the next question is returned
```

Acceptance criteria written as tests using BDD

Continuous Quality IRL (test implementation)


```
68      // Scenario: Submitting an answer to a question
69      @Given("a response of {int} to question {int} for visit {int}")
70      public void a_response_of(Integer responseValue, Integer questionId, Integer visitId) throws JSONException {
71          JSONObject visitObj = new JSONObject();
72          visitObj.put("id", visitId);
73
74          JSONObject questionObj = new JSONObject();
75          questionObj.put("id", questionId);
76
77          JSONObject requestParams = new JSONObject();
78          requestParams.put("visit", visitObj);
79          requestParams.put("question", questionObj);
80          requestParams.put("response", responseValue);
81
82          request = RestAssured.given().body(requestParams.toString());
83      }
84
85      @When("submitting the response")
86      public void submitting_the_response() {
87          response = request.contentType("application/json").post("/answers");
88      }
89
90      @Then("the response is stored and the next question is returned")
91      public void the_response_is_stored_and_the_next_question_is_returned() {
92          response.then().assertThat().statusCode(200).body("sortOrder", equalTo(2));
93      }
94  }
```

Acceptance tests implemented by developer using TDD are executed by the pipeline

Continuous Quality IRL (test results)

company-x-feedback-api

company-x-feedback-api

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
com.companyx.api		91%		n/a	1	4	2	10	1	4
com.companyx.api.controllers		100%		100%	0	21	0	47	0	14
com.companyx.api.models		100%		n/a	0	38	0	20	0	38
com.companyx.api.services		100%		100%	0	6	0	10	0	2
com.companyx.api.viewModels		100%		n/a	0	4	0	4	0	4
Total	5 of 506	99%	0 of 22	100%	1	73	2	91	1	62

Developers receive test results in real time on every commit

company-x-feedback-api > com.companyx.api > ApiApplication

ApiApplication

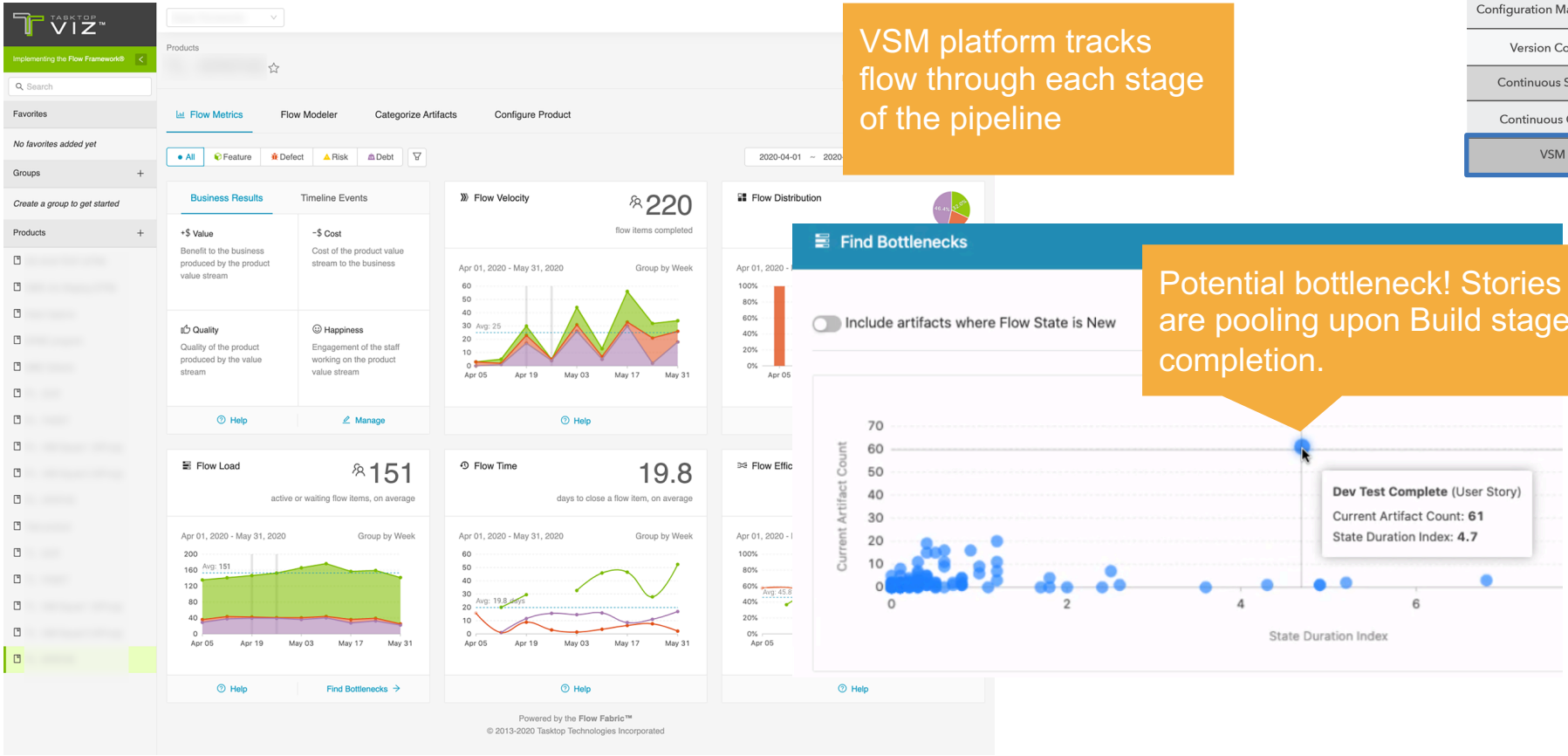
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
main(String[])		0%		n/a	1	1	2	2	1	1
ApiApplication()		100%		n/a	0	1	0	1	0	1
Total	5 of 8	37%	0 of 0	n/a	1	2	2	3	1	2

Value Stream Management (VSM) IRL

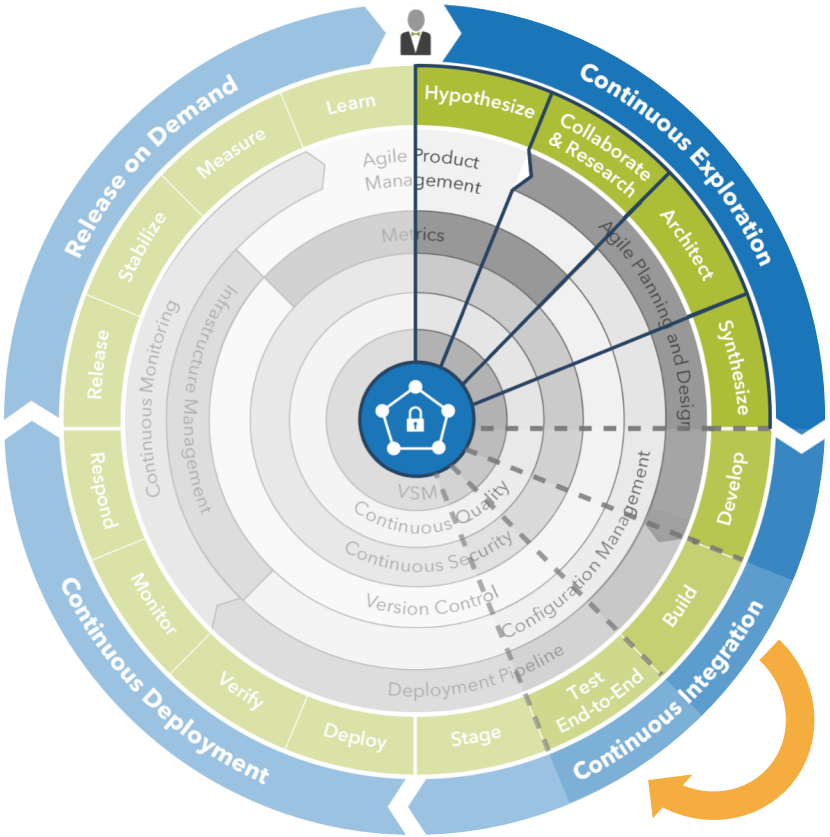
Build
- Deployment Pipeline
- Configuration Management
- Version Control
- Continuous Security
- Continuous Quality
- VSM

VSM platform tracks flow through each stage of the pipeline

Potential bottleneck! Stories are pooling upon Build stage completion.



On to the next step in the CDP



Build

- Deployment Pipeline
- Configuration Management
- Version Control
- Continuous Security
- Continuous Quality
- VSM

Stay tuned

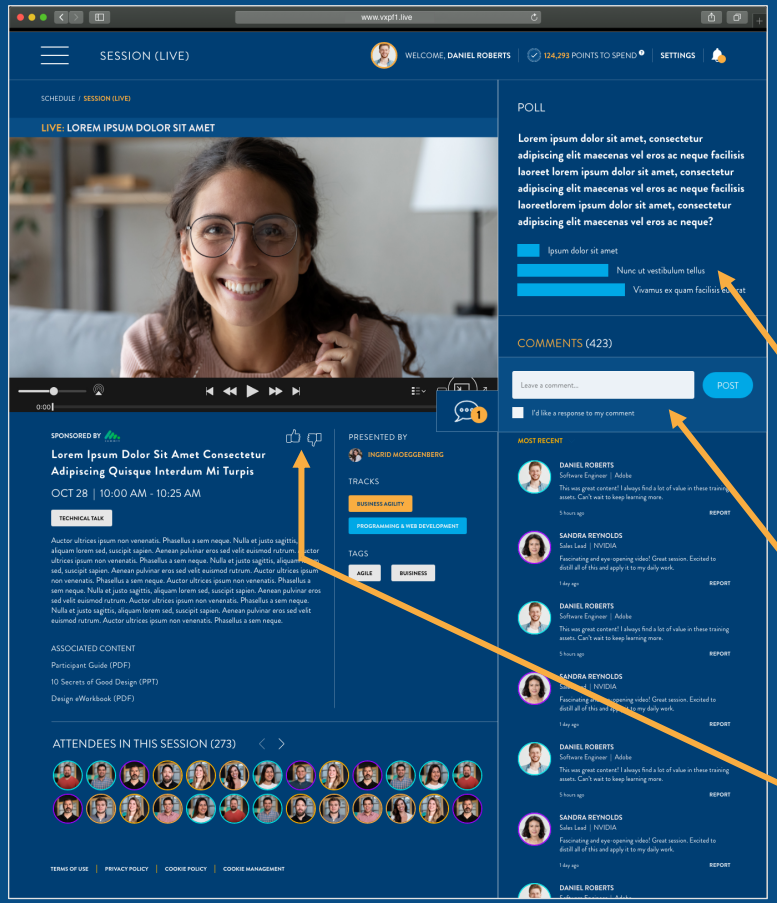
Much more technical DevSecOps guidance is under construction

- Framework content
- Reference architecture
- Technical course

Thank you to Appddiction Studio and TaskTop for providing the examples in this presentation.

Join me at the Meet the Speaker Session!

Please refer to the agenda for scheduled times



Participate in polling, post comments, and rate sessions

1. Polling
2. Comment
3. Thumbs up or down

Thank you!