PRIVACY NOTICE

Your name, company and email address will be shared with Scaled Agile, Inc. to process course fulfillment, which also includes testing and certification. Your information will be used strictly for this purpose in accordance with the Scaled Agile privacy policy available at https://www.scaledagile.com/privacy-policy/.

Table of Contents

Notes:

Notes:

Logistics

- Taking breaks
- Forming social connections during lunch
- Establishing working agreements
- Accessing Wi-Fi
- Locating restrooms

Notes:

Course goals

At the end of this course you should be able to:

- Describe what DevOps is and why it's important
- Describe the importance of continuous testing
- Describe the value of continuous security
- Map your delivery pipeline
- Measure flow through the delivery pipeline
- Identify gaps and delays in flow
- Improve the process of exploring customer needs
- Improve the process of developing, building, and integrating continuously
- Improve the process of deployment to staging and production environments continuously
- Improve the release process
- Build an action plan for a SAFe DevOps transformation

Notes:

Course map

- Lesson 1: Introducing DevOps
- Lesson 2: Mapping Your Pipeline
- Lesson 3: Gaining Alignment with Continuous Exploration
- Lesson 4: Building Quality with Continuous Integration
- Lesson 5: Reducing Time-to-Market with Continuous Deployment
- Lesson 6: Delivering Business Value with Release on Demand
- Lesson 7: Taking Action
- Lesson 8: Becoming a SAFe DevOps practitioner

1-6

Notes:

Lesson 1

Introducing DevOps

Learning objectives:

1.1 Recognize the problem to be solved

1.2 Explain DevOps and its benefits

1.3 Describe continuous security and testing

1.4 Describe core DevOps principles

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

1.1 Recognize the problem to be solved

SCALED AGILE © Scaled Agile, Inc. 1-9

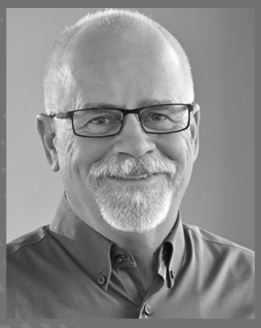
Notes:

Digital disruption is affecting every industry across the globe.

SCALED AGILE © Scaled Agile, Inc.

Notes:

1.1 Recognize the problem to be solved



“Every business is a software business now. Agility isn't an option, or a thing just for teams, it is a business imperative. But we struggle building big systems … ”

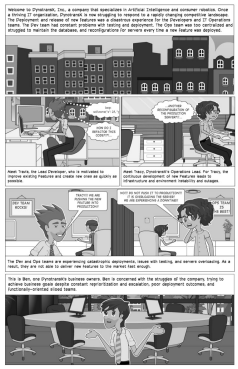
—Dean Leffingwell
Creator of SAFe

SCALED AGILE

Notes:

Chapter 1: The Beginning

- **Step 1:** Read ‘Chapter 1: The Beginning’ in your workbook.
- **Step 2**: Summarize the challenges of the current situation at DynotransX.
- **Step 3:** Discuss as a team:
  - What are some of the challenges you face at your company?
  - Do you agree with any of the solutions offered by Travis and Tracy?



Prepare 10 min Share 5 min

SCALED AGILE © Scaled Agile, Inc.

1-12

Write down the challenges you face at your company:

Do you agree with any of the solutions, offered by Travis and Tracy?

- ☐ Ask the business to invest in infrastructure
- ☐ Create a dedicated on-call team
- ☐ Find a replacement for Travis
- ☐ Establish fast recovery

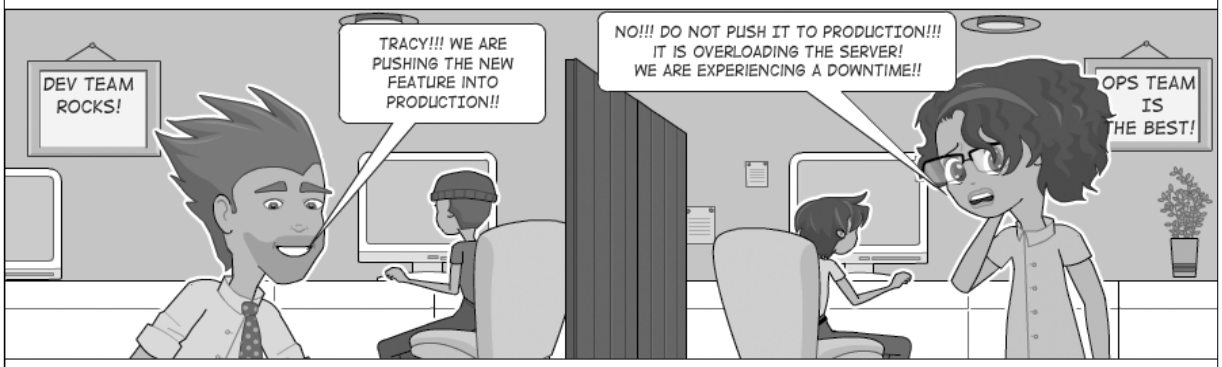
- ☐ Hire more developers and testers
- ☐ Prioritize automation
- ☐ Accelerate work from Dev to Ops
- ☐ Find a replacement for Tracy

Chapter 1: The Beginning

Welcome to DynotransX, Inc., a company that specializes in Artificial Intelligence and consumer robotics. Once a thriving organization, DynotransX is now struggling to respond to a rapidly changing competitive landscape. The deployment and release of new features was a disastrous experience for the Developers and Operations teams. The Dev team had constant problems with testing and deployment. The Ops team was too centralized and struggled to maintain the database, and reconfigurations for servers every time a new feature was deployed.

Meet Travis, the Lead Developer, who is motivated to improve existing features and create new ones as quickly as possible.

Meet Tracy, DynotransX's Operations Lead. For Tracy, the continuous development of new features leads to infrastructure and environment instability and outages.



The Dev and Ops teams are experiencing catastrophic deployments, issues with testing, and servers overloading. As a result, they are not able to deliver new features to the market fast enough.

This is Ben, one DynotransX's business owners. Ben is concerned with the struggles of the company, trying to achieve business goals despite constant reprioritization and escalation, poor deployment outcomes, and functionally–oriented siloed teams.

The problem to be solved

- Every enterprise must be able to quickly validate the wants and needs of customers
- Every enterprise must be able to release Features when customers need them the most

Businesses with traditional software delivery processes struggle to compete effectively in the digital era.

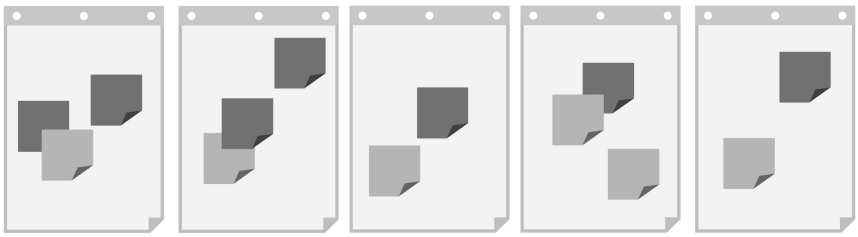
Notes:


1.2 Explain DevOps and its benefits
SCALED AGILE
© Scaled Agile, Inc.
1-14

Notes:

Activity: What does DevOps mean to you?

- **Step 1**: Think about single words or short phrases that define what DevOps means to you.
- **Step 2:** Write each word or phrase (individually or as a team) on a separate sticky note. Hand them to your trainer when you're done.



Prepare

5 min

SCALED AGILE © Scaled Agile, Inc.

1-15

Thought organizer

Video: What is DevOps?

https://vimeo.com/342037390/3a25026214

2 min

SCALED AGILE © Scaled Agile, Inc.

1-16

https://vimeo.com/342037390/3a25026214

Scan me

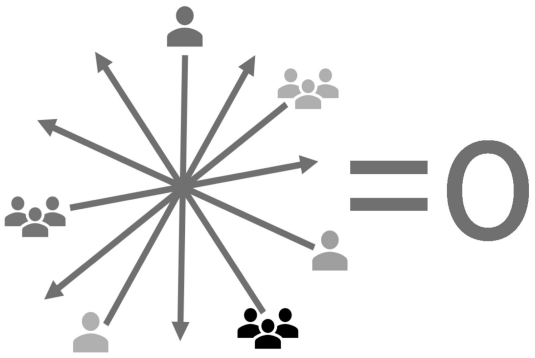
DevOps defined (-ish)

> DevOps is a set of software development practices that combines software development (Dev) and information technology operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.
>
> – Wikipedia

Notes:

Lack of alignment impedes progress

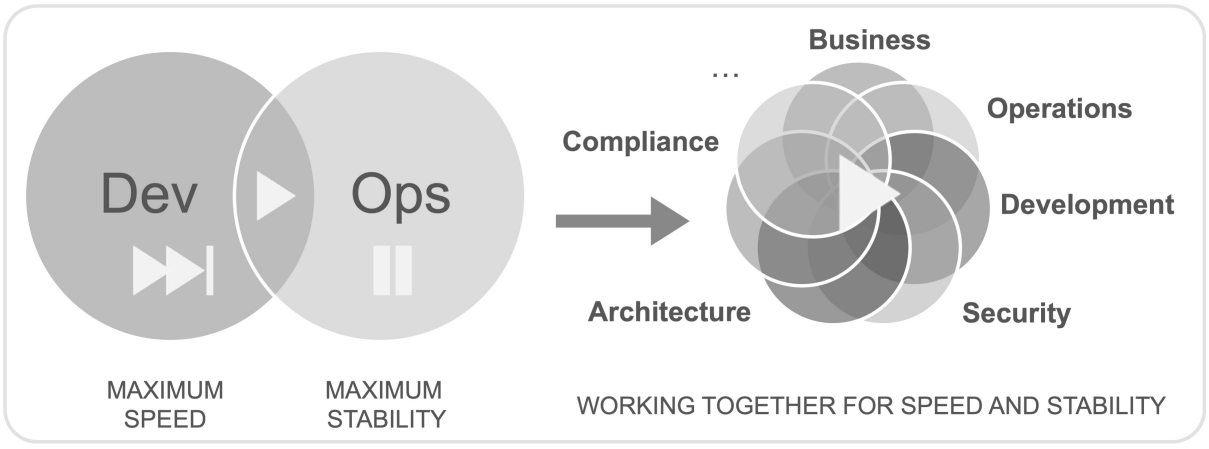
- Different groups in the organization have different goals and direction
- The lack of alignment means their different efforts cancel each other out
- This creates a feeling of constant work with little or no progress



Building alignment is a critical goal of DevOps

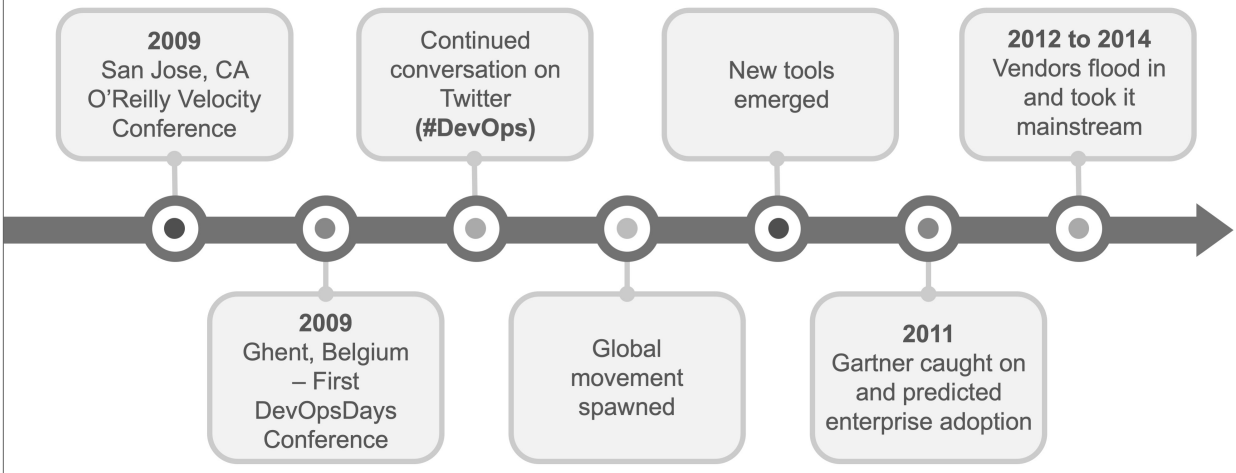
Notes:

Who is DevOps?



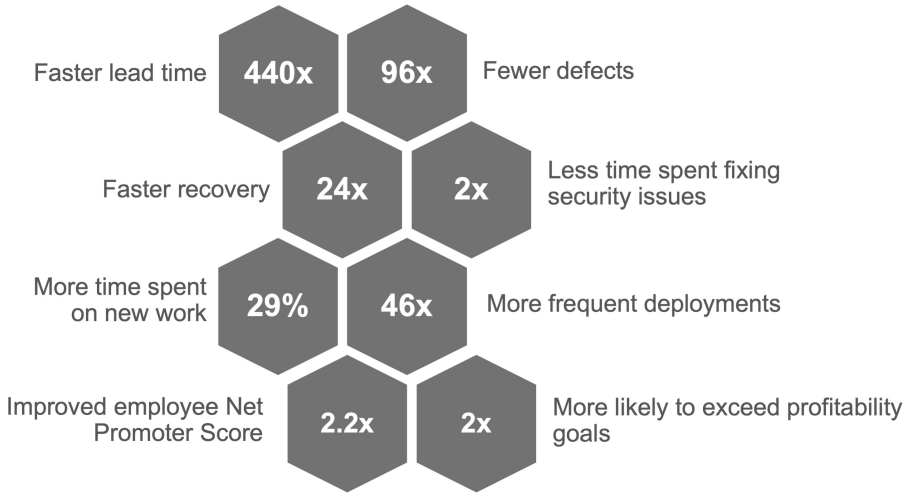
Notes:

The history of DevOps



Notes:

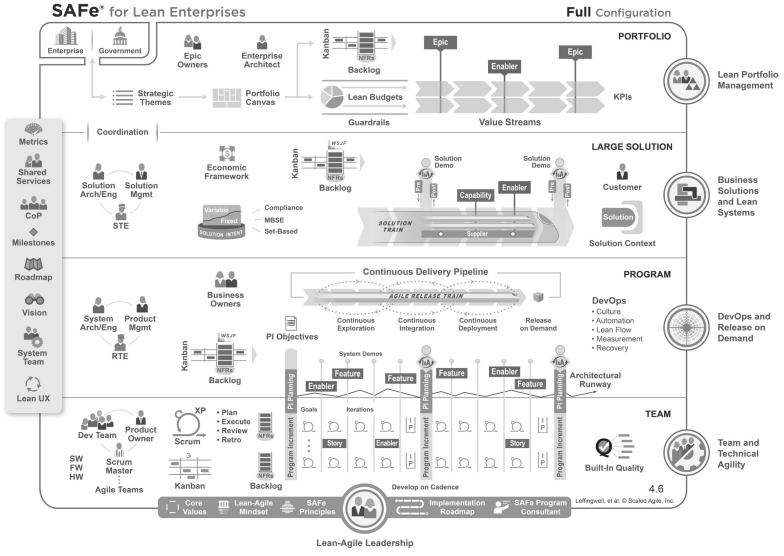
DevOps benefits



Source: https://puppet.com/resources/whitepaper/state-of-devops-report

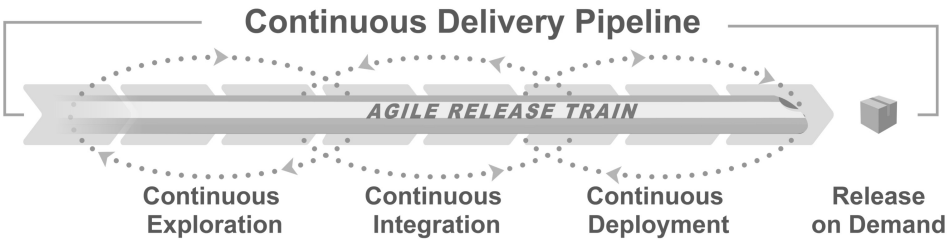
Notes:

SAFe helps bridge the gap between Agile and DevOps



Notes:

The Continuous Delivery Pipeline enables the flow of value



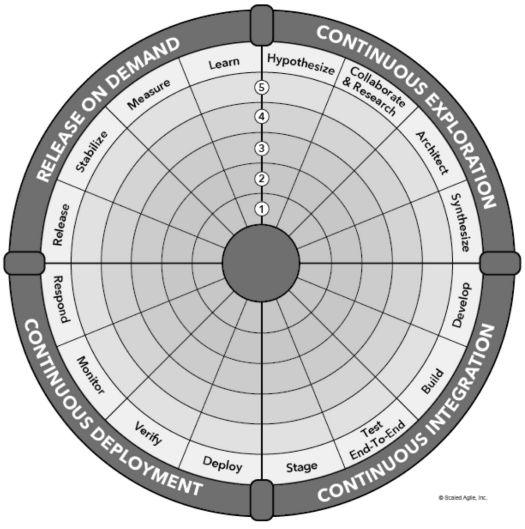
SCALED AGILE © Scaled Agile, Inc. 1-23

Notes:

The DevOps Health Radar helps synchronize the pipeline

Every team can complete a full revolution in less than 24 hours.

– Sr. Engineering Lead, Amazon



SCALED AGILE © Scaled Agile, Inc. 1-24

Notes:

But wait, '*Our problems are different*'

We aren't building a website.

We aren't hosted in the cloud.

Our customers don't want continuous delivery.

Our technology isn't based on micro services.

We don't have just the one platform.

1-25

Notes:

Systems thinking

A common disease that afflicts management the world over is the impression that, "Our problems are different."

They are different to be sure, but the principles that will help to improve quality of product and service are universal in nature.

—W. Edwards Deming

1-26

Notes:

Video: DevOps in a legacy world

As enterprises grow over time, they inevitably add new applications and services, adopt new technology stacks, and still have to deal with legacy applications and systems. But DevOps in the legacy world is NOT a mission impossible.

https://youtu.be/8s46nFCzbck

1 min

SCALED AGILE © Scaled Agile, Inc.

1-27

https://youtu.be/8s46nFCzbck

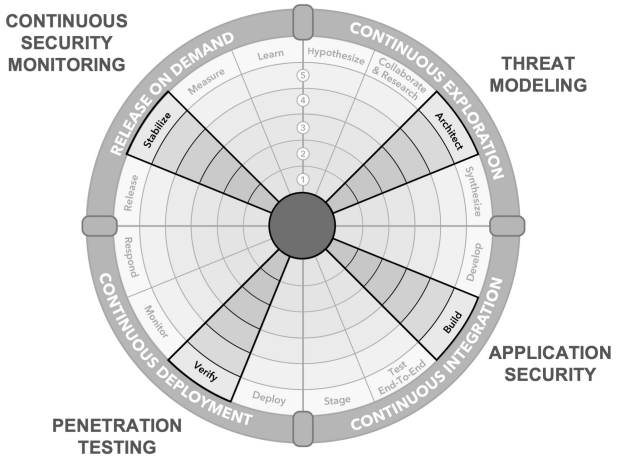
Scan me

1.3 Describe continuous security and testing

Notes:

The role of continuous security

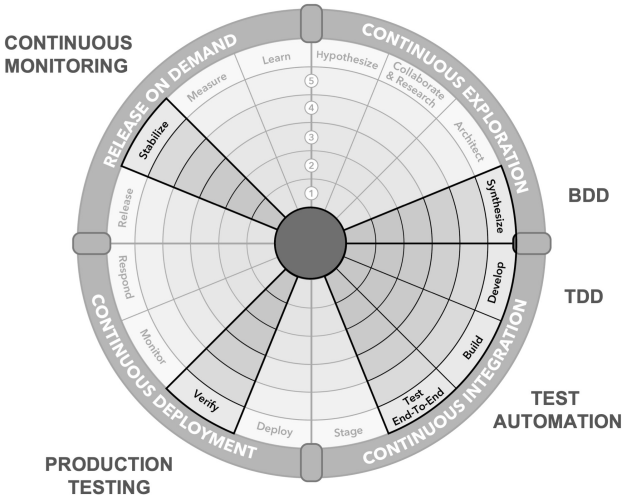
- Information security is a key concept in SAFe DevOps
- DevSecOps concepts present key principles for value delivery
- Security affects every dimension of the continuous delivery cycle
- Information security should be part of every DevOps transformation



Notes:

The role of continuous testing

- Testing is an ongoing activity
- We build quality in by addressing testing and quality throughout the continuous delivery cycle
- Automated testing and quality assurance should be part of every DevOps transformation



1-30

Notes:

1.4 Describe core DevOps principles

Notes:

A CALMR approach to DevOps

- **C**ulture Establish a culture of shared responsibility for development, deployment, and operations.
- **A**utomation Automate the Continuous Delivery Pipeline.
- **L**ean flow Keep batch sizes small, limit WIP, and provide extreme visibility.
- **M**easurement Measure the flow through the pipeline. Implement full-stack telemetry.
- **R**ecovery Architect and enable low-risk releases. Establish fast recovery, fast reversion, and fast fix-forward.

Notes:

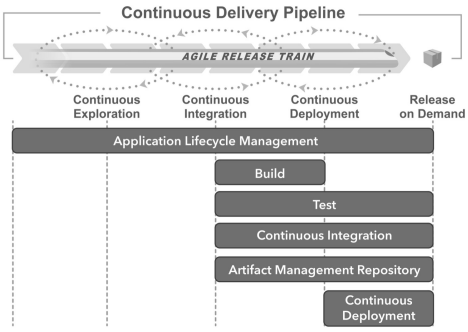
DevOps is a Cultural shift

- Adopt a culture of shared responsibility for development and deployment
- DevOps requires a tolerance for failure and rapid recovery, and rewards risk taking
- Sharing discoveries, practices, tools, and learning across silos is encouraged

Notes:

Automate the deployment process

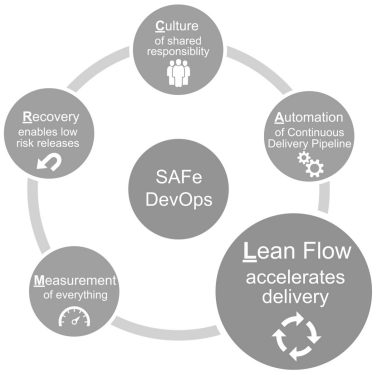
- Automation is a key concept in DevOps
- Automate as much as you can from the Continuous Delivery Pipeline – fewer manual steps improves the flow of value
- Build a comprehensive toolchain to help teams release value more frequently
- Focus on automating healthy processes. If the underlying process is broken, fix it before automating it



Notes:

Focus on continuous Lean flow of value

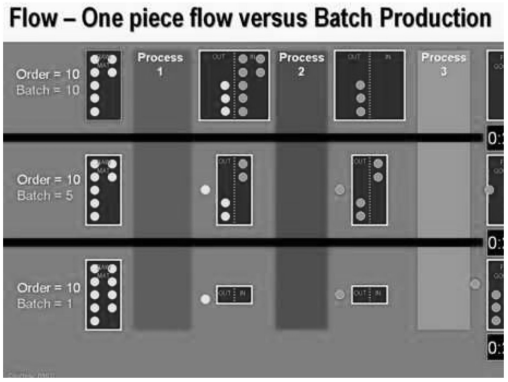
- Identify bottlenecks to the flow of value
- Decrease the batch sizes of the work
- Manage and reduce queue lengths



1-35

Notes:

Video: One piece flow versus batch production



https://youtu.be/JoLHKSE8sfU

1 min

SCALED AGILE © Scaled Agile, Inc.

1-36

https://youtu.be/JoLHKSE8sfU

Scan me

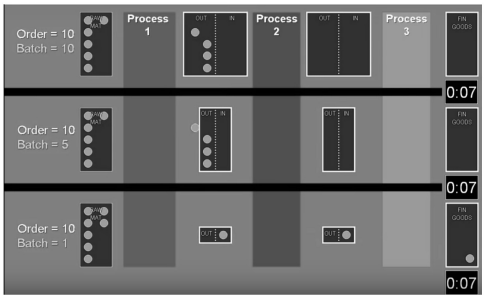
Discussion: Flow

- **Step 1:** At your team, considering the flow of work represented on the diagram, discuss:
  - How do you think the PO for Team 3 feels? How do you think the PO for Team 1 feels?
  - What is a large batch in your context and how can you improve flow?
- **Step 2:** Be prepared to share with the class

PO Team 1

PO Team 2

PO Team 3



Discuss 3 min

Share 2 min

1-37

Thought organizer

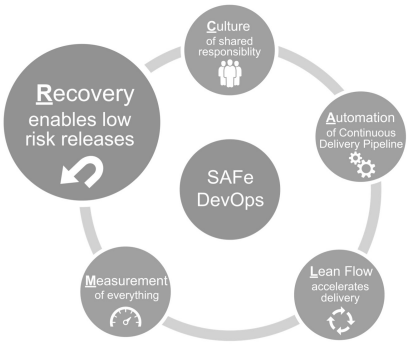
Measure everything

- Collect data on business, application, infrastructure, and client layers
- Collect data about the deployment pipeline itself
- Maintain different telemetry for different stakeholders
- Broadcast measurements
- Continuously improve telemetry during and after problem solving

Notes:

Architect for release-ability and Recovery

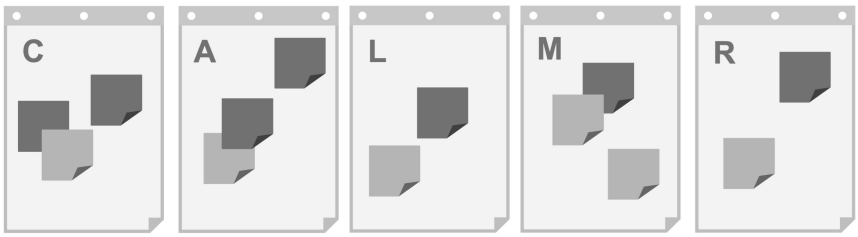
- Adopt a stop-the-line mentality
- Plan for and rehearse failures
- Build the environment for both roll-back and fix-forward



Notes:

Activity: CALMR

- **Step 1:** Revisit the CALMR board
- **Step 2:** Discuss and summarize:
  - Which CALMR concepts are strongly represented?
  - Which ones are under represented? Why?



Summarize

5 min

1-40

Thought organizer

Lesson review

In this lesson you:

- Recognized the problem to be solved
- Explained DevOps and its benefits
- Described continuous security and testing
- Described core DevOps principles

1-41

Notes:

Key Learnings & Insights

Lesson 2

Mapping Your Pipeline

Learning objectives:

2.1 Explain the purpose of mapping the Value Stream

2.2 Visualize the current state of the delivery pipeline using key metrics

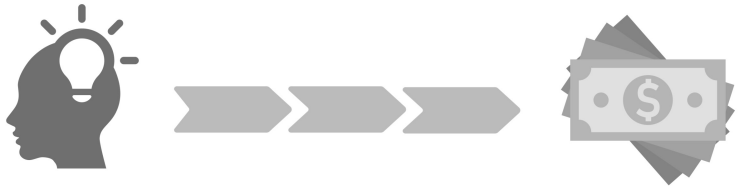
SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

2.1 Explain the purpose of mapping the Value Stream

Notes:

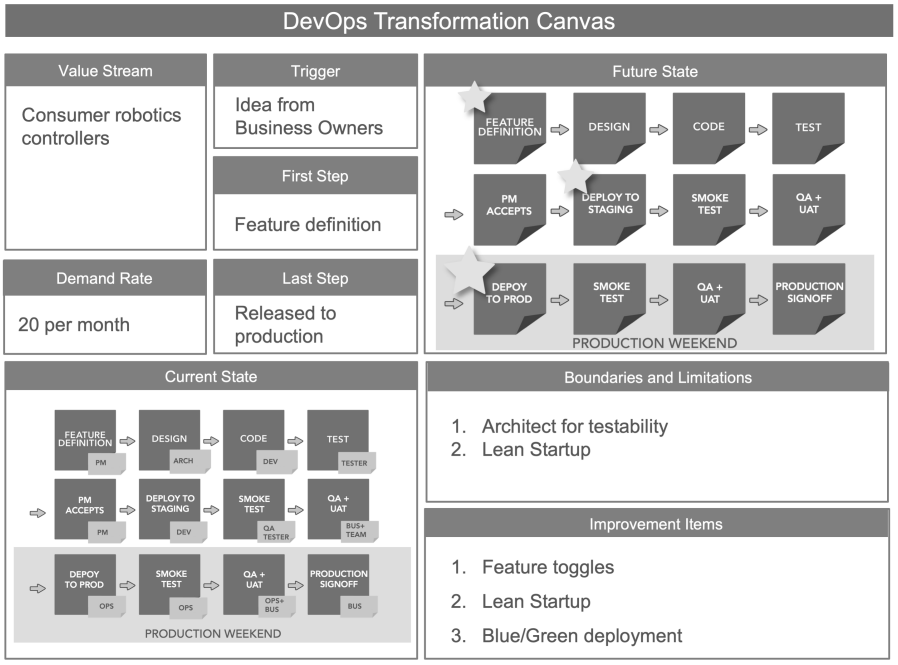
Why map the Value Stream?

- Understand how work flows through the organization from concept to cash
- Measure process quality and organizational efficiency
- Identify bottlenecks to the flow of value
- Understand how we can improve the flow of value



Notes:

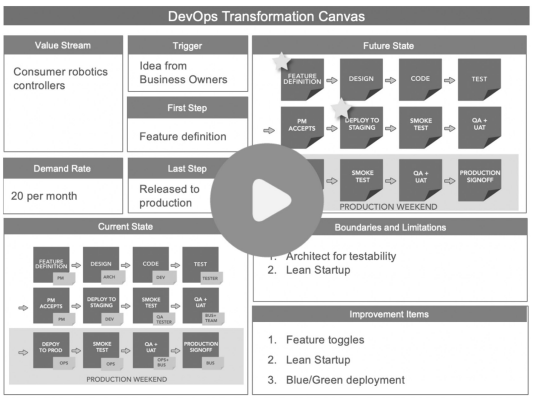
2.1 Explain the purpose of mapping the Value Stream



Notes:

2.1 Explain the purpose of mapping the Value Stream

Video: Value Stream Canvas



https://vimeo.com/325445925/59913369d4

3 min

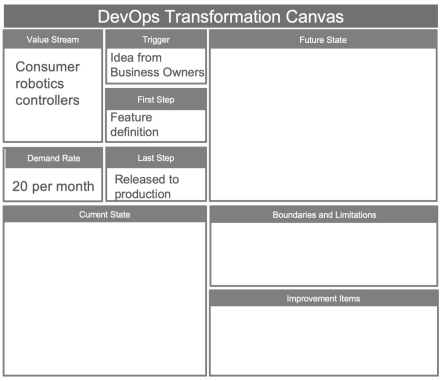
2-6

https://vimeo.com/325445925/59913369d4

Scan me

Problem-Solving: Identify the context for Value Stream mapping

- **Step 1:** On a flip chart or a whiteboard, draw the DevOps Transformation Canvas.
- **Step 2:** Start filling out the canvas by identifying which Value Stream to improve.
- **Step 3:** Identify the the following components:
  - Trigger
  - First Step
  - Last Step
- **Step 4:** What is the Demand Rate?



Prepare

10 min

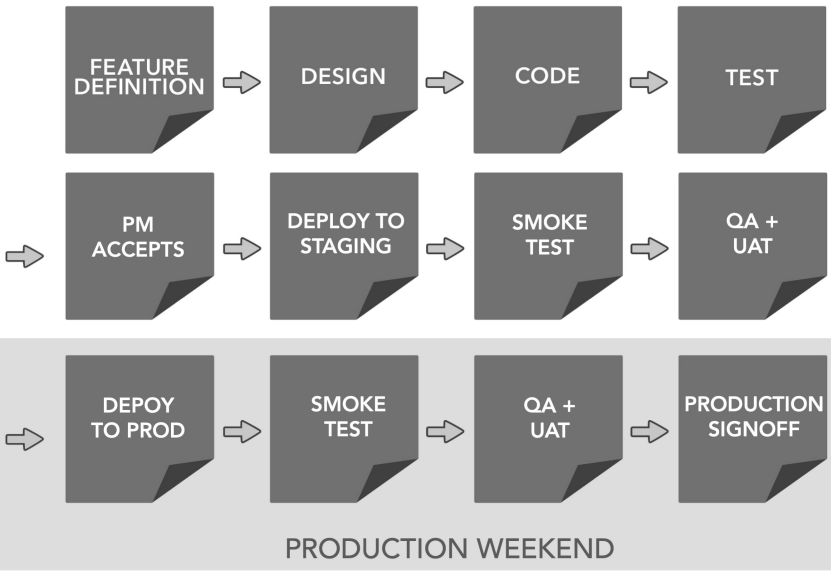
2-7

Thought organizer

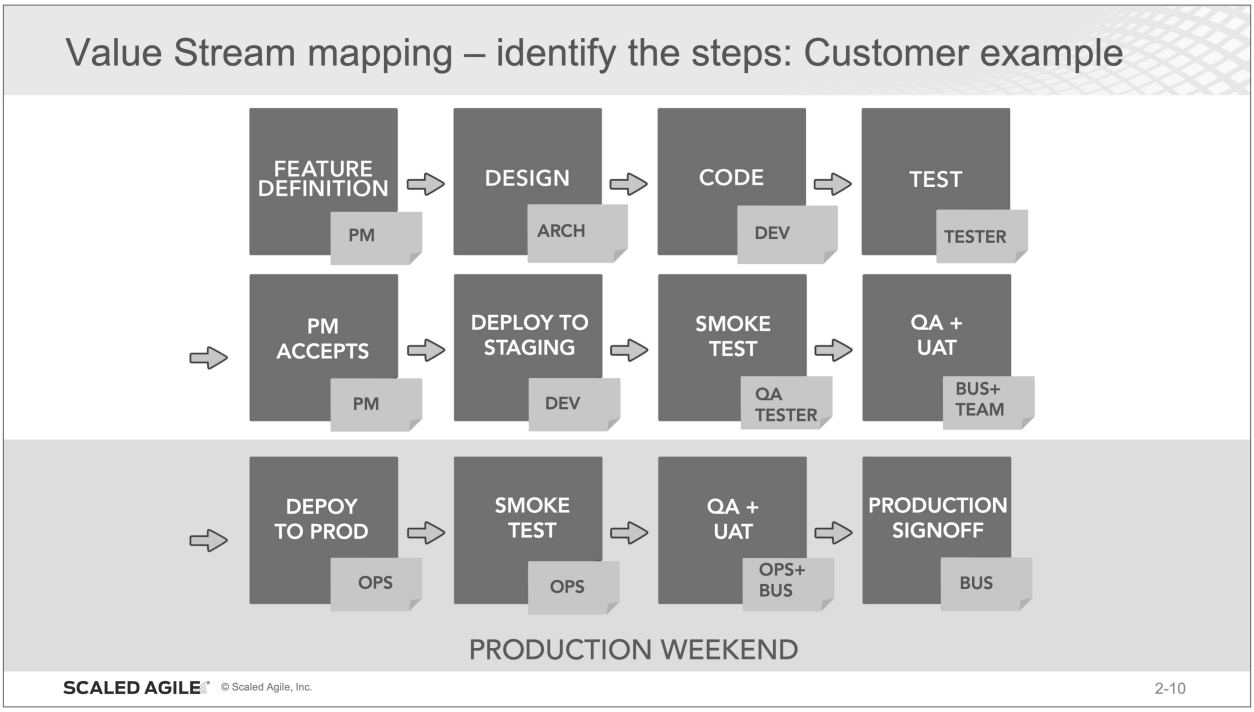
2.2 Visualize the current state of the delivery pipeline using key metrics

Notes:

Value Stream mapping – identify the steps: Customer example



Notes:


Value Stream mapping – identify the steps: Customer example
FEATURE DEFINITION
PM
DESIGN
ARCH
CODE
DEV
TEST
TESTER
PM ACCEPTS
PM
DEPLOY TO STAGING
DEV
SMOKE TEST
QA TESTER
QA + UAT
BUS+ TEAM
DEPOY TO PROD
OPS
SMOKE TEST
OPS
QA + UAT
OPS+ BUS
PRODUCTION SIGNOFF
BUS
PRODUCTION WEEKEND
SCALED AGILE
© Scaled Agile, Inc.
2-10

Notes:

Problem-Solving: Visualize the delivery pipeline

- **Step 1:** On the current state section of the canvas, visualize your current delivery pipeline using sticky notes. Start at the first step and finish at the last step identified previously.
- **Step 2:** Identify the people participating in each step.

Prepare

2-11

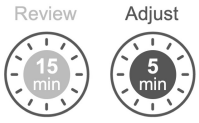
Thought organizer

Problem-Solving: Present your team's map

- **Step 1:** Walk around the room and review other peoples' maps

NOTE: Make sure that at any point there is at least one person to represent your team's map.

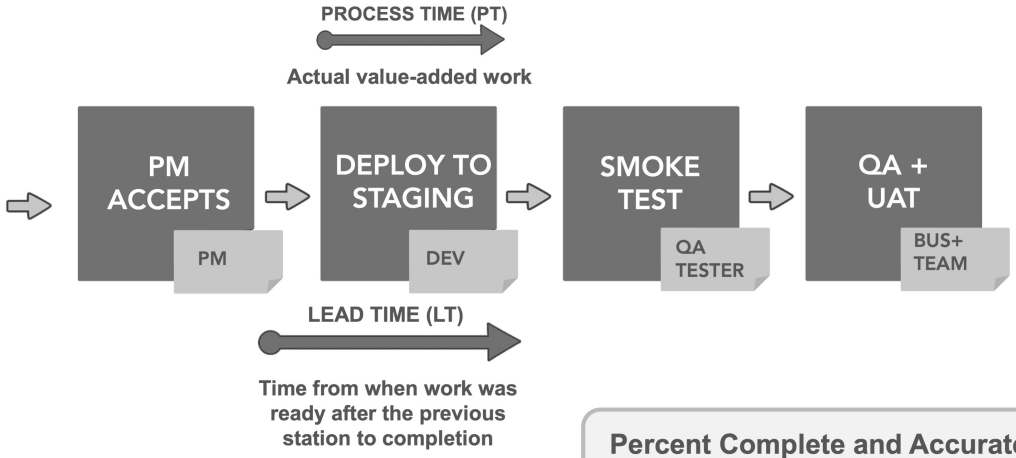
- **Step 2:** Identify items from other maps that you might have missed at your own map
- **Step 3:** As a team, adjust your current state map



2-12

Thought organizer

Value Stream measurements

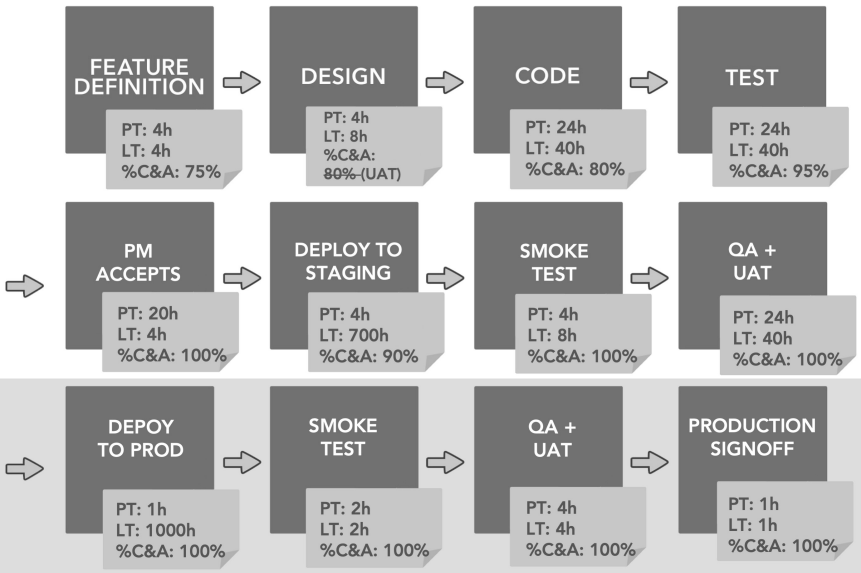


Percent Complete and Accurate (%C&A)

Percent of work that the next station could process as-is

Notes:

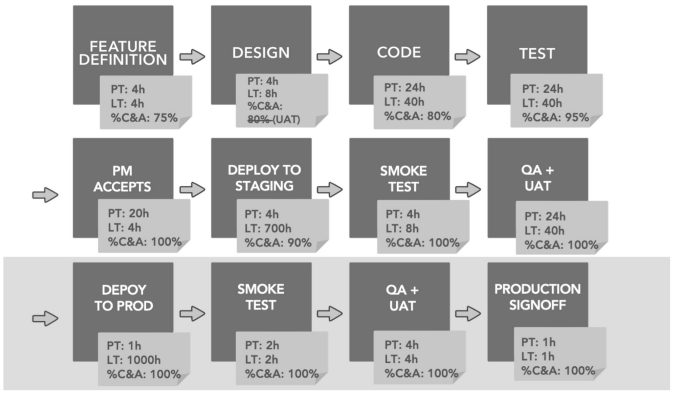
Value Stream mapping: Measure the steps



PRODUCTION WEEKEND

Notes:

Value Stream map: Calculate the total metrics



TOTAL PT = 98 hours
TOTAL LT = 1887 hours
ACTIVITY RATIO = 5%
ROLLED %C&A = 36%

ACTIVITY RATIO = PT/LT
ROLLED %C&A = %C&A 1 * %C&A 2 * %C&A n * 100

Notes:

Problem-Solving: Measure the delivery pipeline

- **Step 1:** Measure the lead time, process time, and % complete and accurate for each step

> HINT: Ask people responsible for the next step about the % complete and accurate of the current step.

- **Step 2:** Calculate the total lead time, total process time, activity ratio, and rolled % complete and accurate

Prepare

30 min

2-16

Thought organizer

Readout: Define and measure the delivery pipeline

- **Step 1:** As a team, present your board
- **Step 2:** Address the following questions:
  - What is the total lead time, total process time?
  - What is the activity ratio and the rolled % complete and accurate?
  - Where are the biggest bottlenecks? Why?

Readout

2-17

Thought organizer

Lesson review

In this lesson you:

- Explained the purpose of mapping the Value Stream
- Visualized the current state of the delivery pipeline using key metrics

SCALED AGILE®

2-18

Notes:

Lesson 2

Mapping Your Pipeline

Key Learnings & Insights

Lesson 3

Gaining Alignment with Continuous Exploration

Learning objectives:

3.1 Create solution hypotheses

3.2 Collaborate and research customer needs

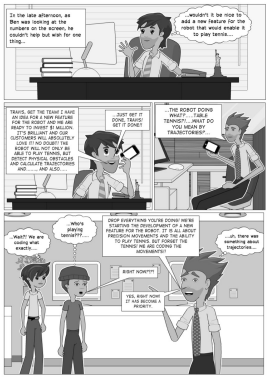
3.3 Architect the solution for continuous delivery

3.4 Synthesize the Vision, the Roadmap, and the Program Backlog

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

Chapter 2: A brilliant idea

- **Step 1:** Read ‘Chapter 2: A brilliant idea’
- **Step 2:** Form pairs and assume the roles of Ben (The Business Owner) and Travis (The Lead Developer)
- **Step 3:** From the perspective of the role you play, defend or challenge the outlined assumptions



Prepare 5 min

Share 5 min

3-3

From the perspective of the role you play, defend or challenge the statements of your character:

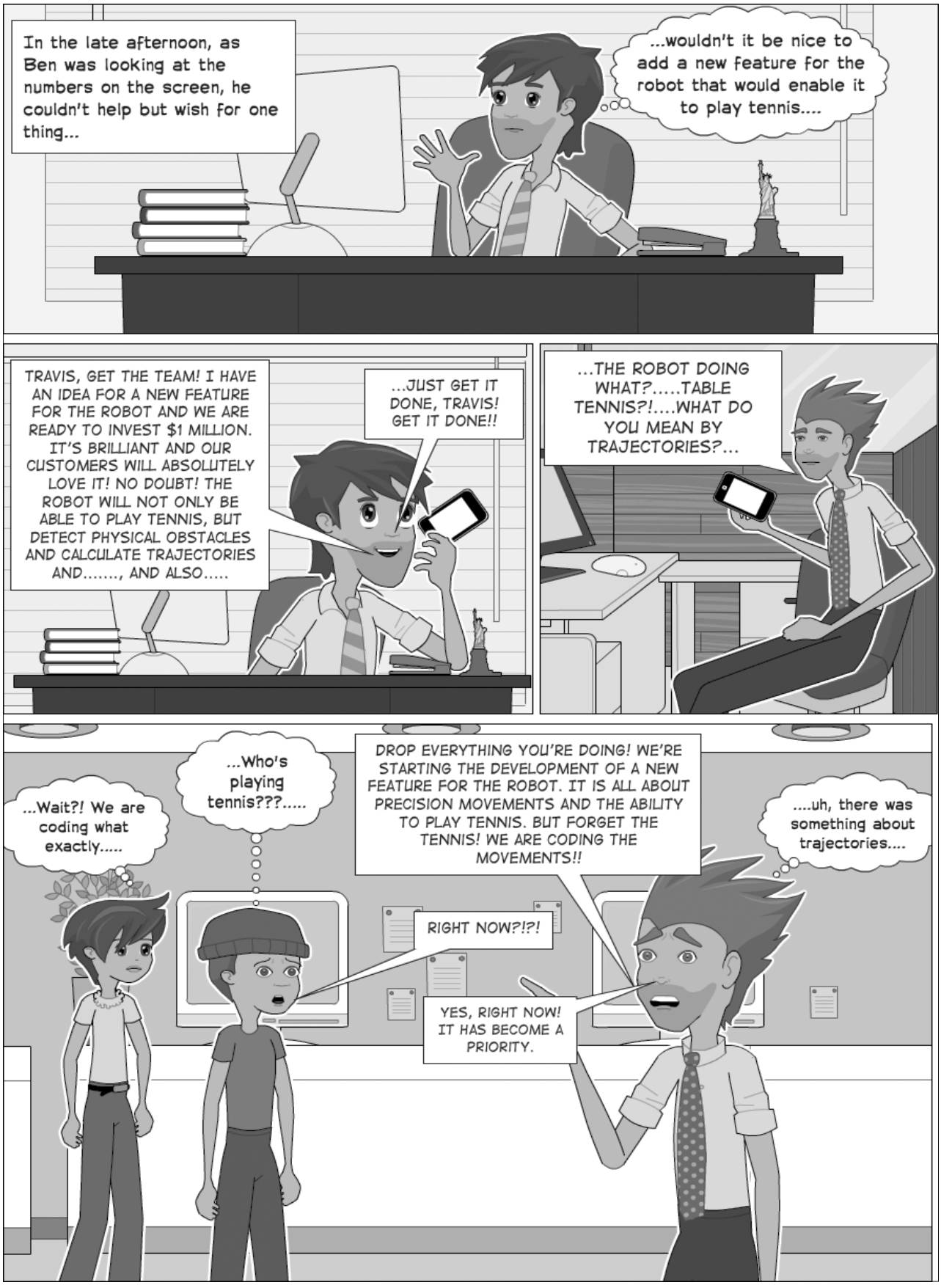
When there is a great idea, it should be prioritized immediately ☐

Only Business Owners decide what work needs to be done ☐

☐ Working on a new feature must start immediately, regardless if it is in the Backlog

☐ There has to be an extensive up-front definition of the feature prior to starting work

Chapter 2: A brilliant idea

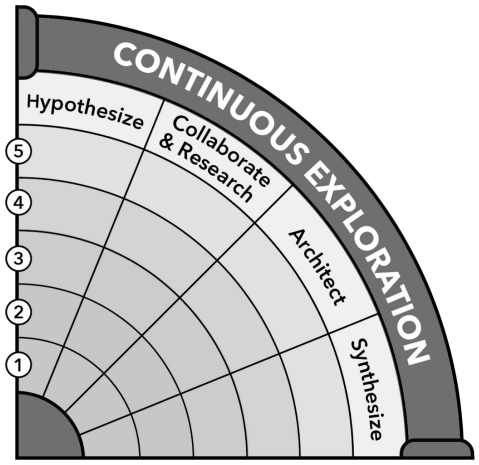


Explore dimension

Business objective (The 'Why')
- Alignment

IT objective (The 'How')
- Continuous Exploration (CE)

Sub-dimensions (The 'What')
- Hypothesize
- Collaborate & Research
- Architect
- Synthesize



Notes:

3.1 Create solution hypotheses

Notes:

Hypothesize

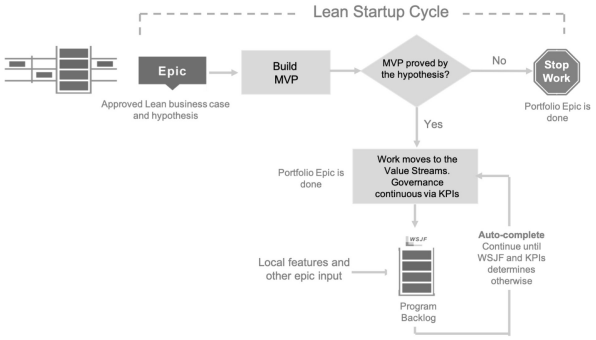
Purpose: Define the hypothesis to be validated through the Continuous Delivery Pipeline

Skills:
- Lean Startup
- Innovation accounting

Notes:

Lean Startup

- The Lean Startup cycle focuses on identifying the viability of ideas
- It follows the Plan-Do-Check-Adjust cycle (PDCA)
- MVP – establish a baseline to test assumptions and gather objective data
- Evaluate hypothesis:
  - If the benefit hypothesis has been proven true, the Value Streams will implement more Features
  - If the hypothesis is proven false, then a decision is made to either pivot with a new hypothesis or to stop work on the Epic



Notes:

Video: Innovation Accounting

https://vimeo.com/332677950/751f8b51dd

2 min

SCALED AGILE © Scaled Agile, Inc.

3-8

https://vimeo.com/332677950/751f8b51dd

Scan me

Innovation accounting

- New products and new features are hard to measure by traditional accounting standards
- When defining an MVP it is important to use metrics that will validate its success or failure
- It is important to focus on metrics that demonstrate real customer engagement and not on vanity metrics

Notes:

Self-assessment: Hypothesize sub-dimension

1. **Sit** — Ideas are vague or not defined.
2. **Crawl** — Ideas are defined (e.g., as epics) but do not include hypothesis statements.
3. **Walk** — Some ideas are expressed as hypothesis statements with measurable outcomes.
4. **Run** — Most ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs.
5. **Fly** — All ideas are expressed as hypothesis statements with measurable outcomes and include an MVPs.

Place a dot on your DevOps Health Radar

Notes:

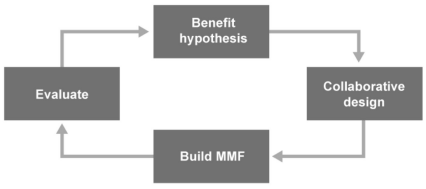
3.2 Collaborate and research customer needs

Notes:

Collaborate & Research

Purpose: Work with multiple stakeholders to understand customer needs

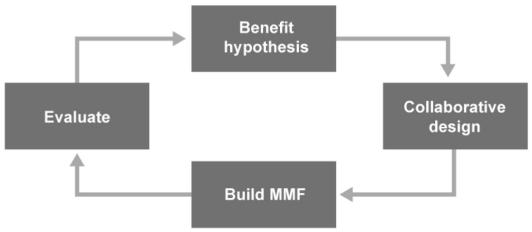
Skills:
- Lean UX
- Research



Notes:

Lean UX

- Lean User Experience (Lean UX) is a mindset, culture, and a process that implements functionality in minimum viable increments and determines success by measuring results against a benefit hypothesis
- Features must be broken into Minimal Marketable Features – the minimum functionality that the teams can build to learn whether the benefit hypothesis is valid or not



Notes:

Research

- **Design Thinking** – Explore the problem and solution spaces using personas, journey maps, empathy maps, and other Agile product management techniques.
- **Customer visits and Gemba walks** – A Gemba walk ('Gemba' is the place where the work is performed) can be used by developers to observe how internal stakeholders execute the steps and specific activities in their operational value streams.
- **Elicitation** – There are a variety of structured elicitation techniques that can be used, such as interviews, surveys, competitive analysis, requirements workshops, and use-case modeling.
- **Trade studies** – Teams engage in trade studies to determine the most practical characteristics of a solution.

Notes:

Self-assessment: Collaborate and research sub-dimension

1. **Sit** — Product Management roles and responsibilities are not defined or followed.
2. **Crawl** — Product Management creates requirements in large batches with little customer or development collaboration.
3. **Walk** — Product Management collaborates with business or development experts, but not both, when defining requirements.
4. **Run** — Product Management regularly collaborates with business, development, and operation experts but does not define Minimum Marketable Features.
5. **Fly** — Product Management always collaborates with business-side, development-side, and operation-side experts and defines Minimum Marketable Features.

Place a dot on your DevOps Health Radar

SCALED AGILE © Scaled Agile, Inc.

3-15

Notes:

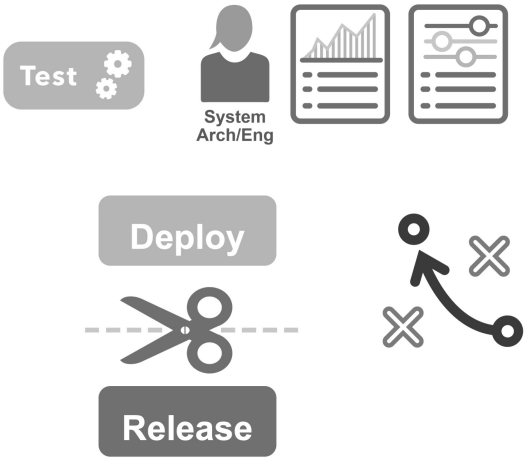
3.3 Architect the Solution for continuous delivery

Notes:

Architect

Purpose: Architect for continuous delivery and DevOps

Skills:
- Architect for testability
- Separate deploy and release
- Decouple release elements
- Architect for operations
- Threat modeling



Notes:

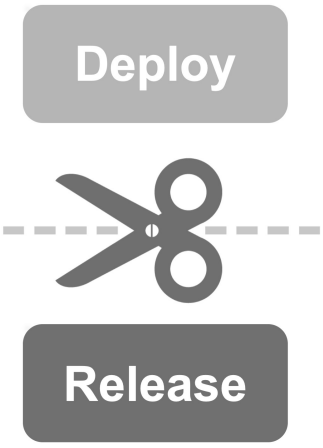
Architect for testability

- Systems that can't readily be tested can't readily be changed
- In a system that is designed for testability, all jobs require less time
- Patterns that can accelerate value flow:
  - Domain Driven Design (DDD)
  - Loose Coupling & API-Driven Architecture
  - Cloud-Native Architecture
  - Microservices & Containerization
  - Serverless Architecture
  - Strangler pattern

Notes:

Separate deploy from release

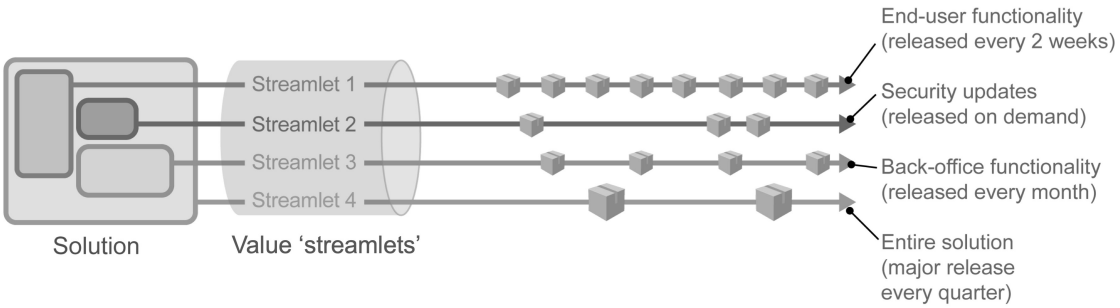
- Separate deploy to production from release
- Hide all new functionality under feature toggles
- Enable the ability to deploy and verify in production and release on demand



Notes:

Decouple release elements

- Different parts of the solution require different release strategies
- Architect the solution to enable the various strategies and to shift them over time based on business demand



Notes:

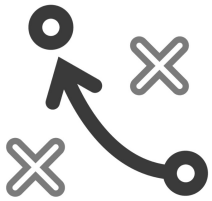
Architect for operations

- Take the operational needs into account
- Build telemetry and logging capabilities into every application and into the solution as a whole
- Allow services to be downgraded or even removed in times of high loads or in response to incidents
- Build capabilities for fast recovery and for fix-forward

Notes:

Threat modeling

- Information security considerations should start early
- Identify potential security threats and attack vectors
- Architect to address security concerns
- Ensure backlogs reflect important security requirements



Notes:

Self-assessment: Architect sub-dimension

1. Sit	—	Architecture is monolithic and fragile; it is difficult to change and involves managing complex dependencies across many components and systems.
2. Crawl	—	Architecture is predominantly monolithic but some applications/systems are loosely coupled.
3. Walk	—	Architecture is most decoupled but doesn't allow Release on Demand.
4. Run	—	Architecture is aligned around value delivery with few dependencies across components and systems.
5. Fly	—	Architecture is built for Release on Demand and operability.

Place a dot on your DevOps Health Radar

1 min

Notes:

3.4 Synthesize the Vision, the Roadmap, and the Program Backlog

Notes:

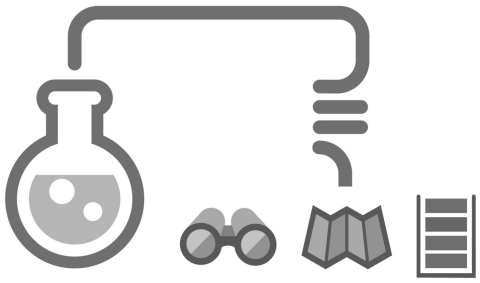
Synthesize

Hypothesize | Collaborate & Research | Architect | **Synthesize**

Purpose: Synthesize the hypotheses, research, collaboration, and architecture into a vision, a roadmap, and backlog. Gain alignment with PI planning.

Skills:

- Feature writing
- Behavior-Driven Development (BDD)
- Economic prioritization
- PI Planning



Notes:

Feature writing

- 'Feature' is an industry-standard term familiar to marketing and Product Management
- Benefit hypothesis justifies Feature implementation cost, and provides business perspective when making scope decisions
- Acceptance criteria are typically defined during Program Backlog refinement
- Reflects functional and Nonfunctional Requirements (NFRs)
- Fits in one Program Increment

SSO example:

Cruise Control & Speed finder

Benefit hypothesis

Cruise Control and Speed finder will allow us to sell *50% more cars*

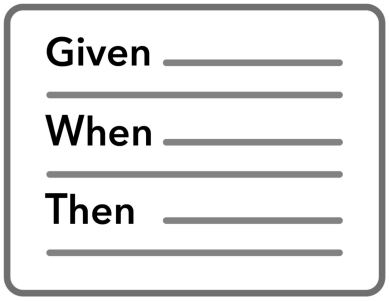
Acceptance criteria

Given Vehicle is traveling with cruise control at the posted speed limit
When a new speed limit is detected
Then vehicle automatically adjusts to the new speed limit

Notes:

Behavior-Driven Development (BDD)

- Behavior-driven development is a test-first approach to writing requirements
- Gherkin syntax (Given-When-Then) is commonly used
- Permits building executable specifications
- Acceptance criteria become executable tests



Notes:

Economic prioritization

- In a flow system, job sequencing is the key to economic outcomes.
- Give preference to jobs with *shorter Duration* and *higher Cost of Delay (CoD)*, using *Weighted Shortest Job First* (WSJF)
- WSJF provides a way of understanding the cost of delay and focusing on items that provide the best cost of delay reduction in the shortest time

$$\text{WSJF} = \frac{\text{Cost of Delay}}{\text{Job Duration (Job size)}}$$

Notes:

PI Planning

Program Increment (PI) Planning is a cadence-based, face-to-face event that serves as the heartbeat of the Agile Release Train (ART), aligning all the teams on the ART to a shared mission and Vision.

- Two days every 8 – 12 weeks (10 weeks is typical)
- Everyone attends in person if at all possible
- Product Management owns Feature priorities
- Development teams own Story planning and high-level estimates
- Architect/Engineering and UX work as intermediaries for governance, interfaces, and dependencies

PI Planning

Agile Team

Notes:

Self-assessment: Synthesize sub-dimension

1. Sit — The program backlog does not exist or is not shared.

2. Crawl — The program backlog exists, but the features are incomplete and prioritization is an afterthought.

3. Walk — The program backlog contains fully defined features, but are not prioritized using WSJF.

4. Run — Features in the program backlog are complete, prioritized using WSJF and matched to the delivery capacity of the ART.

5. Fly — The program backlog is a collection of minimum marketable features created using BDD and prioritized using WSJF.

Place a dot on your DevOps Health Radar

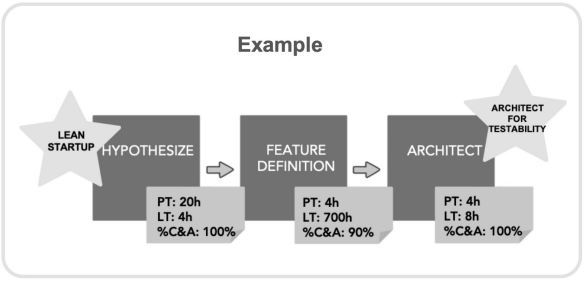
SCALED AGILE © Scaled Agile, Inc.

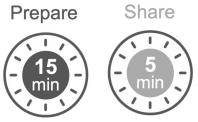
3-30

Notes:

Activity: Apply Continuous Exploration to your Value Stream

- **Step 1:** Identify the skills in this lesson that would improve flow in your value stream between idea capture and ready to code. Write each on a star-shaped sticky note.
- **Step 2:** Construct a new view of this part of the value stream in the Future State portion of the DevOps Transformation Canvas. (Reuse, reorder, or replace steps as desired.)
- **Step 3:** Place the stars on the steps they would enable and provide new Process Time, Lead Time, and %C&A for each step.





3-31

Thought organizer

Lesson review

In this lesson you:

- Explored how to create solution hypotheses
- Discussed how to collaborate and research customer needs
- Explored how to architect the solution for continuous delivery
- Explored how to synthesize the Vision, the Roadmap, and the Program Backlog

3-32

Notes:

Lesson 3

Gaining Alignment with Continuous Exploration

Key Learnings & Insights

Lesson 4

Building Quality with Continuous Integration

Learning objectives:

4.1 Develop the solution

4.2 Build continuously

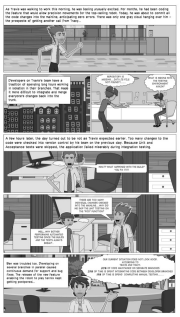
4.3 Test end-to-end

4.4 Validate on a staging environment

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

Chapter 3: Continuous Integration problems

- **Step 1:** Read 'Chapter 3: Continuous Integration problems.'
- **Step 2:** In your group, discuss common challenges with Continuous Integration (CI).
- **Step 3:** In one statement, summarize the biggest challenge with CI in your own context.
- **Step 4:** Discuss as a team:
  - Do you share similar CI challenges with other members in your group?



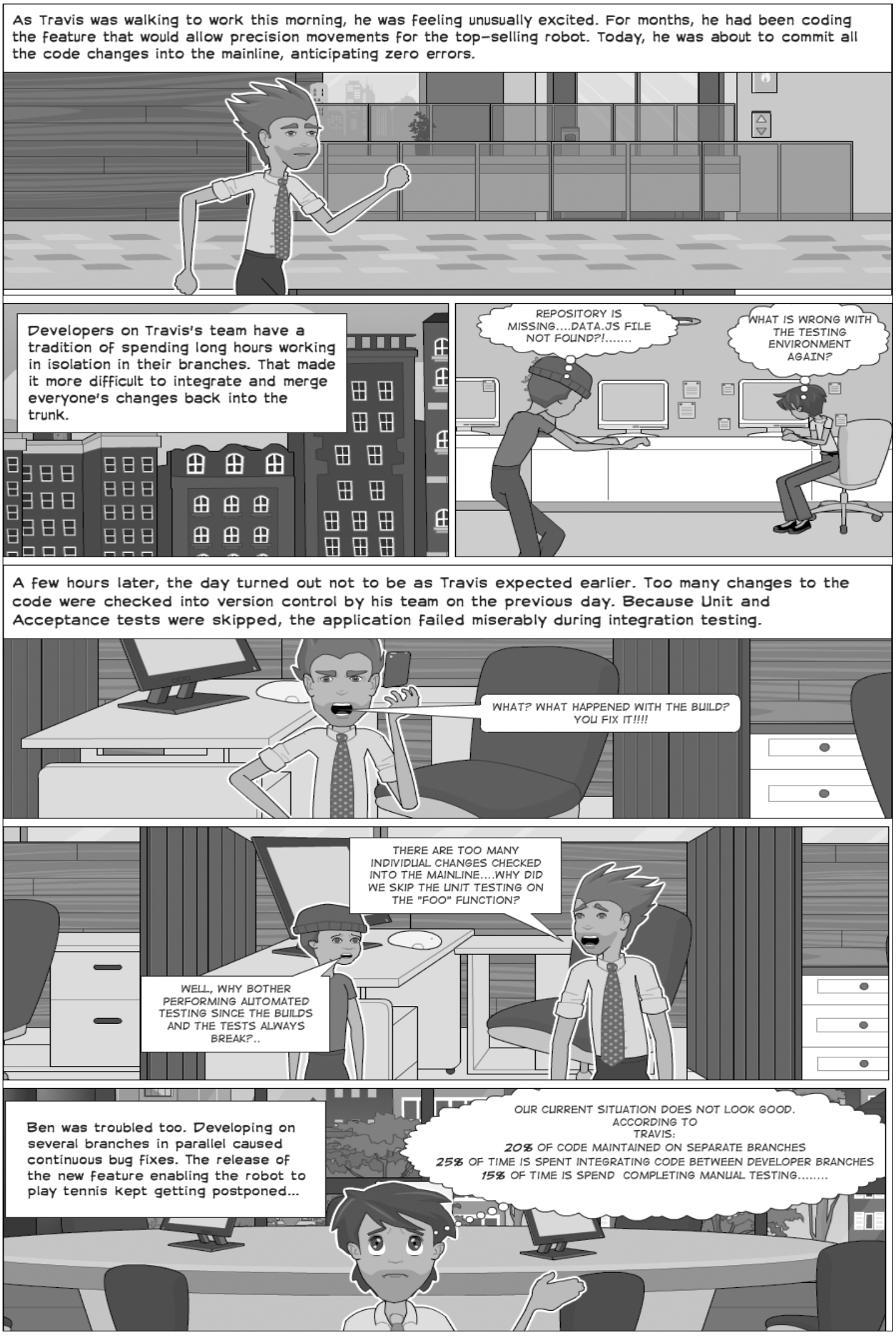
Prepare 5 min Share 5 min

Summarize the biggest challenge with CI in your own context:

Do you share similar CI challenges with other members in your group?

Are any of the options below, possible solutions towards CI?

- ☐ Frequent code commits, larger batch sizes
- ☐ Less frequent code commits, small batches
- ☐ Establish gated commits
- ☐ Adopt trunk-based development to predict higher throughput

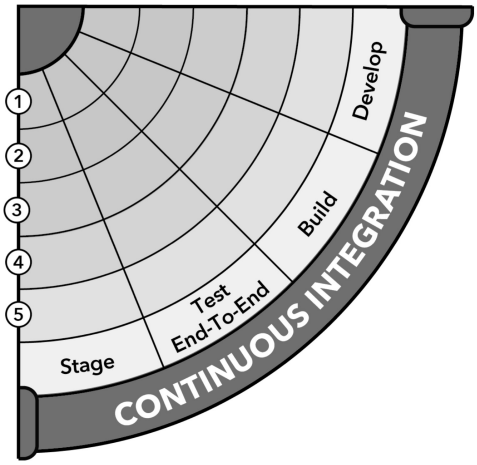

As Travis was walking to work this morning, he was feeling unusually excited. For months, he had been coding the feature that would allow precision movements for the top-selling robot. Today, he was about to commit all the code changes into the mainline, anticipating zero errors.
Developers on Travis's team have a tradition of spending long hours working in isolation in their branches. That made it more difficult to integrate and merge everyone's changes back into the trunk.
REPOSITORY IS MISSING....DATA.JS FILE NOT FOUND?!.....
WHAT IS WRONG WITH THE TESTING ENVIRONMENT AGAIN?
A few hours later, the day turned out not to be as Travis expected earlier. Too many changes to the code were checked into version control by his team on the previous day. Because Unit and Acceptance tests were skipped, the application failed miserably during integration testing.
WHAT? WHAT HAPPENED WITH THE BUILD? YOU FIX IT!!!!
THERE ARE TOO MANY INDIVIDUAL CHANGES CHECKED INTO THE MAINLINE....WHY DID WE SKIP THE UNIT TESTING ON THE "FOO" FUNCTION?
WELL, WHY BOTHER PERFORMING AUTOMATED TESTING SINCE THE BUILDS AND THE TESTS ALWAYS BREAK?..
Ben was troubled too. Developing on several branches in parallel caused continuous bug fixes. The release of the new feature enabling the robot to play tennis kept getting postponed...
OUR CURRENT SITUATION DOES NOT LOOK GOOD. ACCORDING TO TRAVIS:
20% OF CODE MAINTAINED ON SEPARATE BRANCHES
25% OF TIME IS SPENT INTEGRATING CODE BETWEEN DEVELOPER BRANCHES
15% OF TIME IS SPEND COMPLETING MANUAL TESTING.....

Integrate dimension

Business objective (The 'Why')
- Quality

IT objective (The 'How')
- Continuous Integration (CI)

Sub-dimensions (The 'What')
- Develop
- Build
- Test end-to-end
- Stage



Notes:

4.1 Develop the Solution

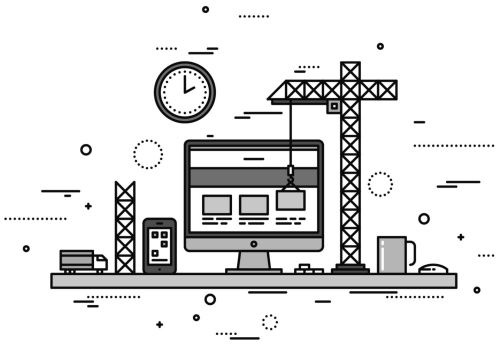
Notes:

Develop

Purpose: Implement a Story or a part of a Story and commit the code

Skills:

- Break features into stories
- Test-Driven Development
- Version control
- Engineering practices
- Pair work
- Application telemetry
- Threat modeling (covered in Architect)



Notes:

Break Features into Stories

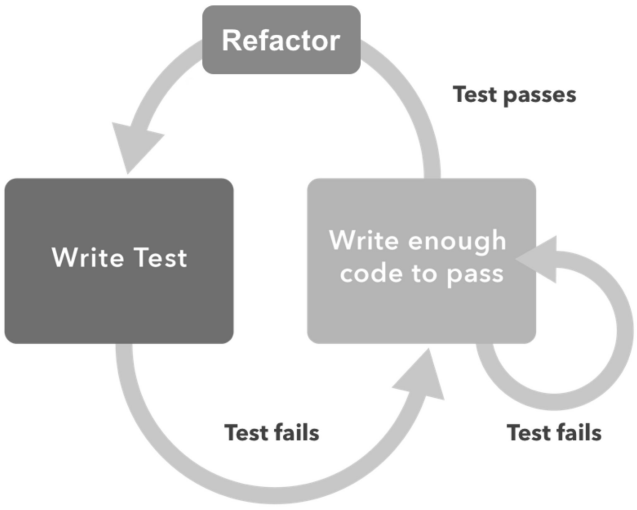
Techniques for splitting Features and Stories to fit within their boundaries (PI and Iteration, respectively).

1. Workflow steps
2. Business rule variations
3. Major effort
4. Simple/complex
5. Variations in data
6. Data methods
7. Defer system qualities
8. Operations
9. Use-case scenarios
10. Break out a spike

 4-7

Notes:

Test-Driven Development (TDD)



 4-8

Notes:

Version control

- Maintain all assets under version control
- From requirements, to code, to configuration, to tests, and test data
- Establish clear check-in and check-out procedures
- Version control improves traceability for automating compliance

Notes:

Agile Software Engineering

Modern engineering practices boost speed and quality. They include:

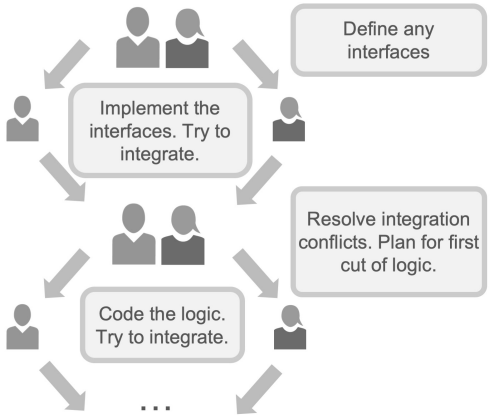
- Test-first mindset
- Agile modeling
- Emergent design
- Pattern-based coding
- Code reviews

Notes:

Pair work

Pair work improves system quality, design decisions, knowledge sharing, and team velocity.

- Pair work is …
  - A collaborative effort of any two team members: dev/dev, dev/PO, dev/tester, etc.
  - Broader and less constraining than pair programming
- Team members spend 20% to 80% time pairing
- Spontaneous pairing, and purposeful rotation over time



Example user Story implementation flow

Notes:

Application telemetry

Application telemetry enables faster identification of problems from production incidents

- Telemetry should cover all levels of the code — from methods, to components, to services, to the entire application
- Application design must take into account operational health telemetry
- Features must include the ability to measure the benefit hypothesis against both leading and trailing indicators

Notes:

Self-assessment: Develop sub-dimension

1. **Sit** — The team backlog does not exist or is not used to manage daily work.
2. **Crawl** — Stories are either incomplete or too verbose; unit tests are generally not written; peer reviews are not conducted.
3. **Walk** — Stories are complete; most changes have unit tests; peer reviews are usually conducted.
4. **Run** — Code is checked in daily; unit test coverage is 80%+; peer reviews are always conducted..
5. **Fly** — Code is checked in multiple times per day; tests are written before code (TDD); pair work and other Built-in quality practices are the norm.

Place a dot on your DevOps Health Radar

SCALED AGILE © Scaled Agile, Inc.

4-13

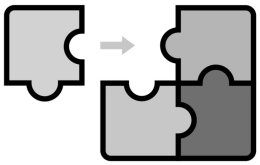
Notes:

4.2 Build continuously

Notes:

Build

Purpose: Compile source files into deployable binaries, verify that code functions as the developer(s) intended & merge dev branches to trunk

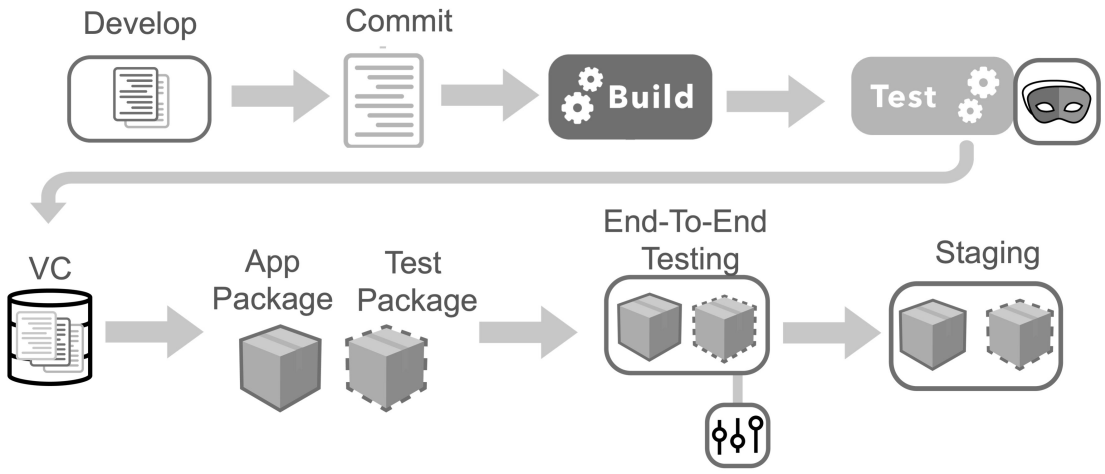


Skills:

- Continuous code integration
- Build and test automation
- Trunk-based development
- Gated commit
- Application security

Notes:

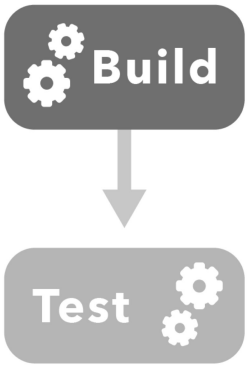
Continuous code integration



Notes:

Build and Test automation

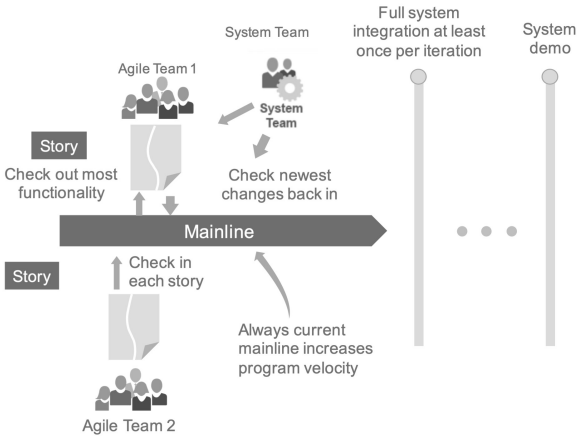
- Initiate a build often, preferably on every commit
- Run unit tests as part of the build
- Run static code analysis as part of the build
- Visualize and monitor the build and test process
- Report failures immediately
- Broken builds are the highest priority



Notes:

Trunk-based development

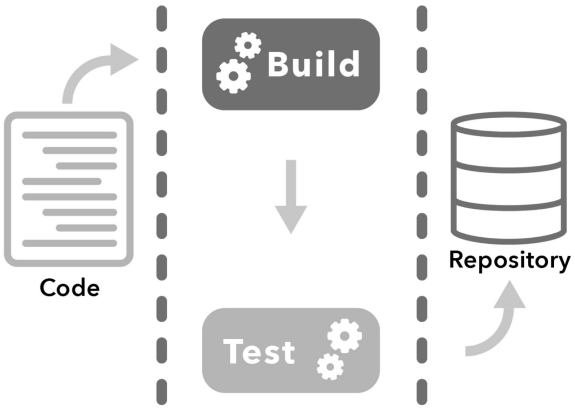
- Single trunk/main for all teams
- Each commit merges to main
- Avoid long-lived branches
- Avoid multiple open branches



Notes:

Gated commit

- Gated commits ensure that broken code doesn't block the rest of the developers or the pipeline
- Only changes that have passed all build and quality checks are committed to version control
- Alert the code author immediately upon rejection



Notes:

Application security

- Apply tools to automatically identify security vulnerabilities in the code during the build process
- Assess open source libraries continuously for known vulnerabilities to identify risks during development or build processes

Notes:

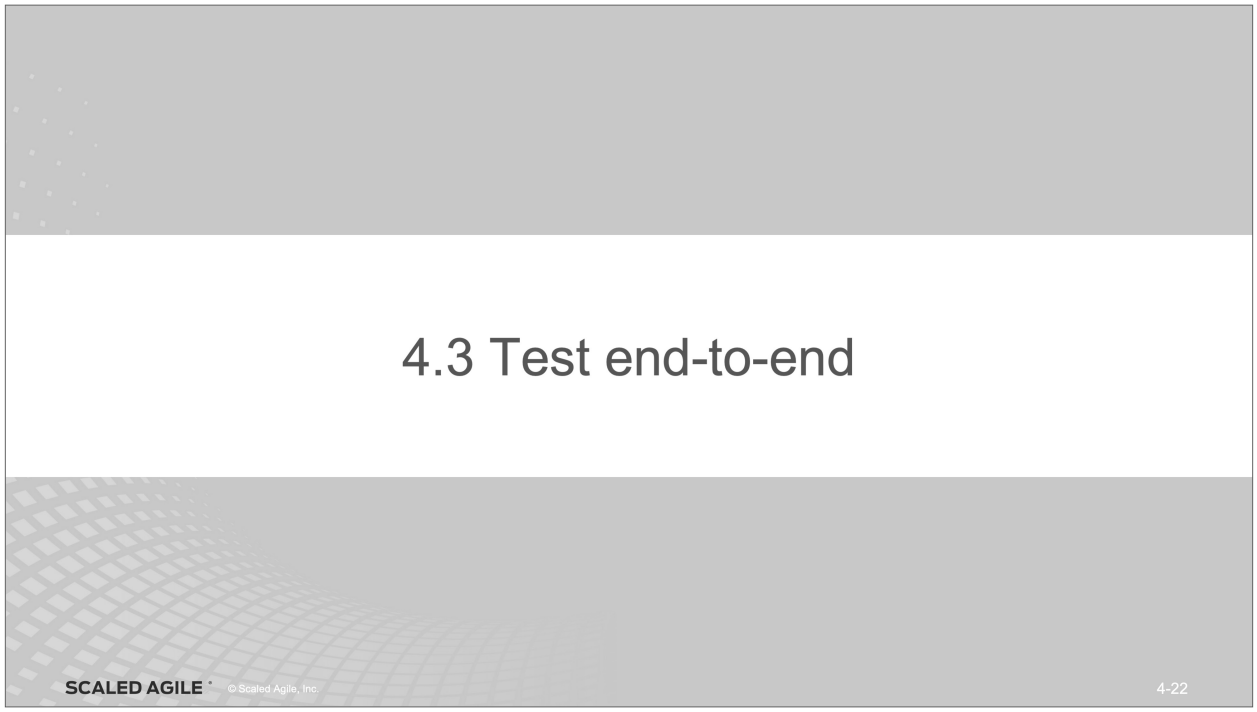
Self-assessment: Build sub-dimension

1. Sit	—	Builds are run fewer than once per iteration and/or are completely manual.
2. Crawl	—	Builds are run once per iteration and are partially automated; dev branches are open for a month or more; builds break often.
3. Walk	—	Automated builds run once a day; broken builds are corrected in 2-4 hours; manual unit tests are run against each build; dev branches are open for 2-4 weeks.
4. Run	—	Builds run automatically upon code commit; broken builds are corrected within 1 hour; automated unit tests are run against each build; dev branches are merged to trunk every iteration.
5. Fly	—	Builds run on every commit; builds include static code analysis and security testing; gated commits prevent defects from entering the version control; dev branches are merged to trunk on every commit.

Place a dot on your DevOps Health Radar

1 min

Notes:



Notes:

Video: Bing - Continuous Delivery

https://youtu.be/3sFT7tgyEQk

4 min

SCALED AGILE © Scaled Agile, Inc.

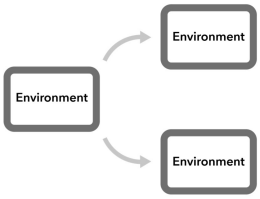
4-23

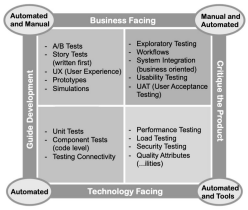
https://youtu.be/3sFT7tgyEQk

Scan me

Test end-to-end

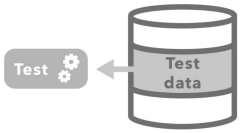
Purpose: To validate changes against acceptance criteria in an integrated, production-simulated environment





Skills:

- Test and production environment congruity
- Test automation
- Test data management
- Service virtualization
- Nonfunctional requirements (NFRs)



Notes:

Test and production environment congruity

At Telstra, only 50% of the source code in their development and test environments matched what was running in production.

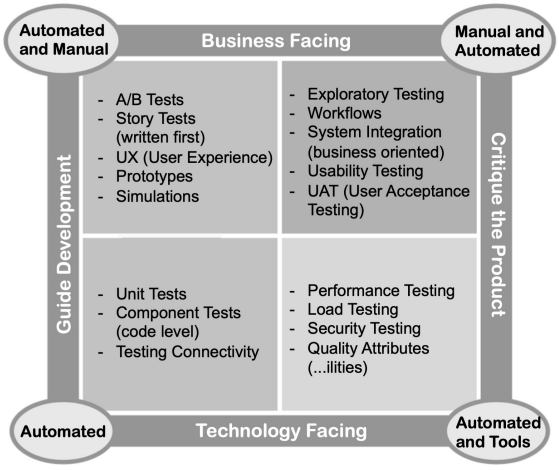
– Em Campbell-Pretty

- Make sure the test environments match production as much as possible
- Maintain all configuration changes under version control
- Service virtualization helps elevate some cost considerations
- Invest in higher fidelity for more accurate testing

Notes:

Test automation

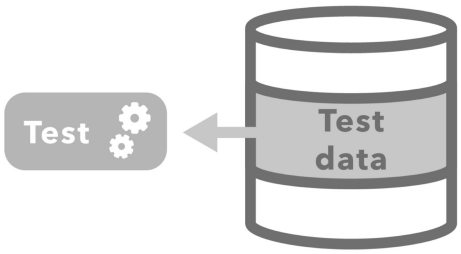
- Many types of testing need to be run:
  - Functional testing
  - Integration testing
  - Regression testing
  - Performance testing
  - Security testing
  - Exploratory testing
  - Penetration testing
- Not all tests should be automated



Notes:

Test data management

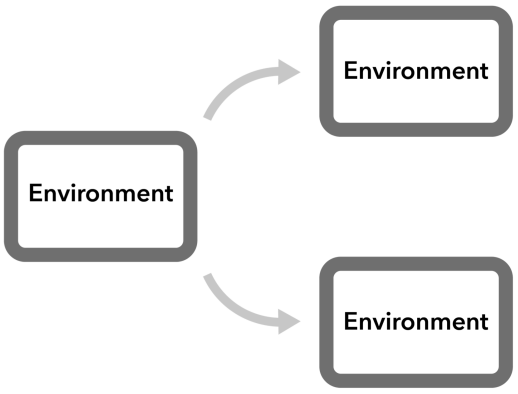
- Data for all types of tests must be managed
- Store data in a repository for consistent testing
- Emulate production data to ensure tests reflect realistic situations



Notes:

Service virtualization

- Ability to spawn environments which match production to test
- Environments that support different types of testing
- Maintain environment data in source control



Notes:

Nonfunctional Requirements (NFRs)

Nonfunctional Requirements are system qualities that support end-user functionality and system goals.

- Sometimes known as the 'ilities'— reliability, usability, scalability, availability, etc.
- NFRs are *constraints* on backlog items
- NFRs are not backlog items themselves

Notes:

Self-assessment: Test end-to-end sub-dimension

1. **Sit** — Testing is performed manually in environments that do not mimic production; testing occurs in large batches during a scheduled "testing" phase
2. **Crawl** — Testing is mostly manual in non-production-like environments; stories are implemented and tested independently within a single PI
3. **Walk** — Half the testing is automated and performed in production-like, or production-simulated, environments every PI
4. **Run** — The majority of tests are automated and run in production-like environments; stories are implemented and fully tested every iteration
5. **Fly** — Successful builds trigger automatic deployment to production-like test environments; all tests are automated; tests run in parallel and changes are fully validated after every commit

Place a dot on the DevOps Health Radar

SCALED AGILE © Scaled Agile, Inc.

4-30

Notes:

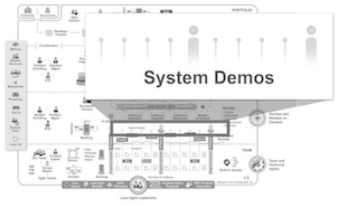
4.4 Validate on a staging environment

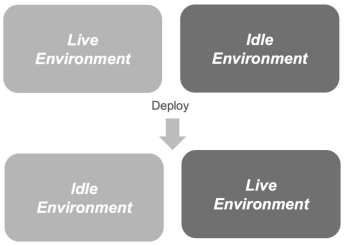
Notes:

Stage

Purpose: Host fully-validated, systems in a production-grade environment, from which they can be deployed to production

Skills:
- Maintain a staging environment
- Blue/Green deployment
- System demo





Notes:

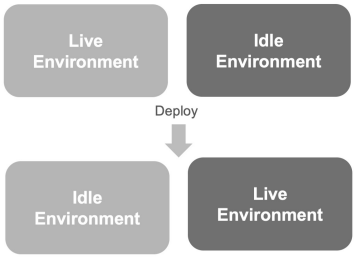
Maintain a staging environment

- Maintain a staging environment that matches production to prepare for moving to production
- Deploy to staging at least every iteration and run your system demos from there
- Ideally, deploy to staging automatically after all build and end-to-end tests have passed

Notes:

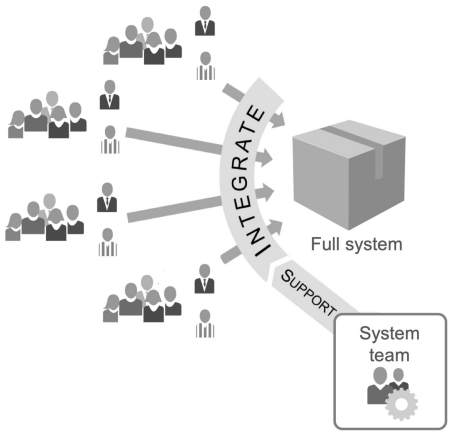
Blue/Green deployment

- Maintain two identical environments: idle and live
- New features are deployed to idle environment on continuous basis. While staged, the features can be tested and showcased in preparation for release
- New code is released to the idle environment, where it is thoroughly tested. When the code is ready to be released the team makes the idle environment active
- If problems are discovered after the switch, the active environment is switched back to idle, restoring the previously active environment
- Switching between environments is typically done by redirecting traffic at the load balancer



Notes:

System demo



- Demo working integrated systems every two weeks
- New Features work together, and with existing functionality
- Demo from a staging environment that resembles production as much as possible
- Program stakeholders provide feedback

4-35

Notes:

Self-assessment: Stage sub-dimension

1. Sit	—	No staging environment exists or we use a test environment for staging.
2. Crawl	—	Features are deployed manually to a staging environment once every PI.
3. Walk	—	Features are deployed to a staging environment once per month and demonstrated to Product Management.
4. Run	—	Features and infrastructure are auto-deployed to a staging environment every iteration and accepted by Product Management.
5. Fly	—	Stories, changes and infrastructure are auto-deployed to a staging environment, and deployment is validated.

Place a dot on your DevOps Health Radar

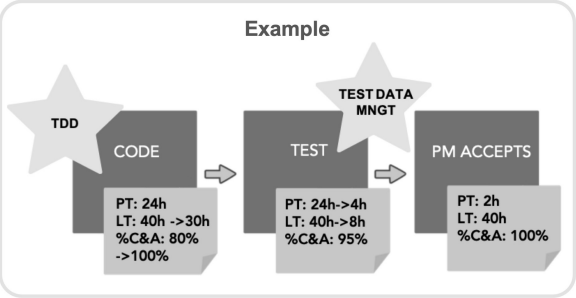
1 min

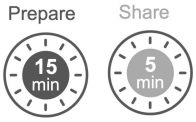
4-36

Notes:

Activity: Apply Continuous Integration to your Value Stream

- **Step 1:** Identify the skills in this lesson that would improve flow in your value stream between ready to code and ready to deploy. Write each on a star-shaped sticky note.
- **Step 2:** Construct a new view of this part of the value stream in the Future State portion of the DevOps Transformation Canvas. (Reuse, reorder, or replace steps as desired.)
- **Step 3:** Place the stars on the steps they would enable and provide new Process Time, Lead Time, and %C&A for each step.





4-37

Thought organizer

Lesson review

In this lesson you:

- Explored how to develop the solution
- Discussed building continuously
- Analyzed testing end-to-end
- Explored how to validate on a staging environment

Notes:

Lesson 4

Building Quality with Continuous Integration

Key Learnings & Insights

Lesson 5

Reducing Time-to-Market with Continuous Deployment

Learning objectives:

5.1 Deploy to production

5.2 Verify the solution

5.3 Monitor for problems

5.4 Respond and recover

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

Chapter 4: Deployment gone wrong

- **Step 1:** Read 'Chapter 4: Deployment gone wrong.'
- **Step 2:** Form pairs and assume the roles of Tracy and Travis.
- **Step 3:** Consider some of the immediate solutions suggested by your character. Could any of them be the right course of action?
- **Step 4:** Explain to your partner the reasons why the option you chose may or may not be an appropriate solution.

Prepare 5 min

Share 5 min

5-3

Considering some of the immediate solutions suggested by your character, could any of them be the right course of action?

- ☐ Build immutable infrastructure
- ☐ Recovery issues are not real issues
- ☐ Add more servers
- ☐ Resolve production issues before they cause business disruption

- ☐ Deploy to select production environments
- ☐ Create two databases: blue and orange
- ☐ Match the staging and prod environments
- ☐ Decouple database changes from app changes

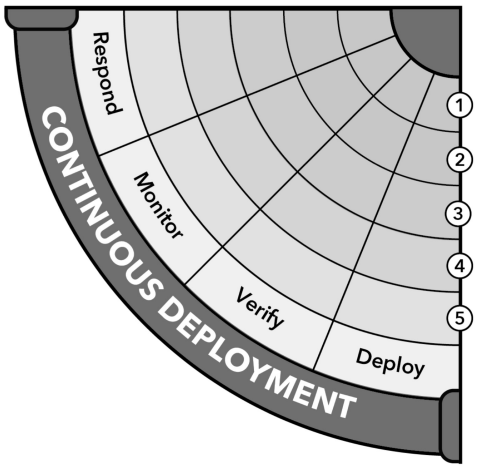
Chapter 4: Deployment gone wrong

Deployment dimension

Business objective (The 'Why')
- Time to market

IT objective (The 'How')
- Continuous Deployment (CD)

Sub-dimensions (The 'What')
- Deploy
- Verify
- Monitor
- Respond



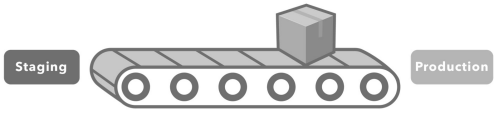
Notes:

5.1 Deploy to production

Notes:

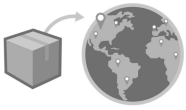
Deploy

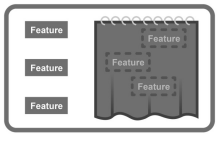
Purpose: Deploy changes into production with high frequency and low risk



Skills:

- Dark launches
- Feature toggles
- Infrastructure as code
- Deployment automation
- Selective deployment
- Self-service deploys
- Version control (covered in Build)
- Blue/Green deployment (covered in Stage)

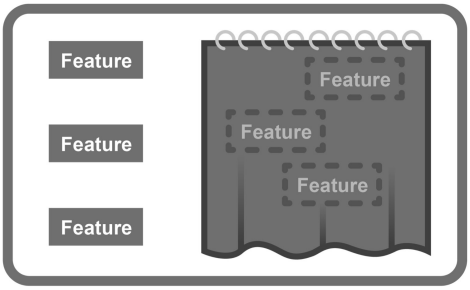




Notes:

Dark launches

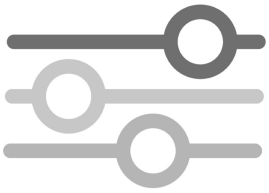
- Separate deploy (to production) from release (to end users)
- Enables testing and monitoring system behavior in the actual production environment before exposing new functionality to users



Notes:

Feature toggles

- Dynamically show and hide features in production
- Enables separation of deploy and release
- Enables rapid rollback of problem features
- Test toggles in both on and off position

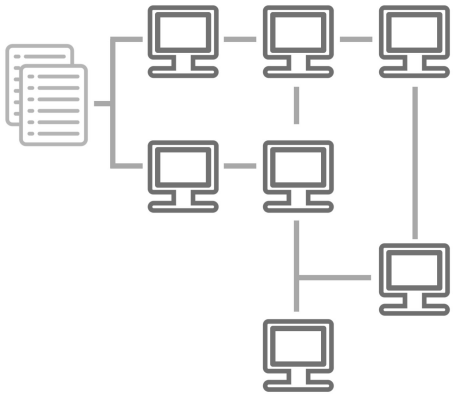


Be careful of toggle overload and testing complexity

Notes:

Infrastructure as code

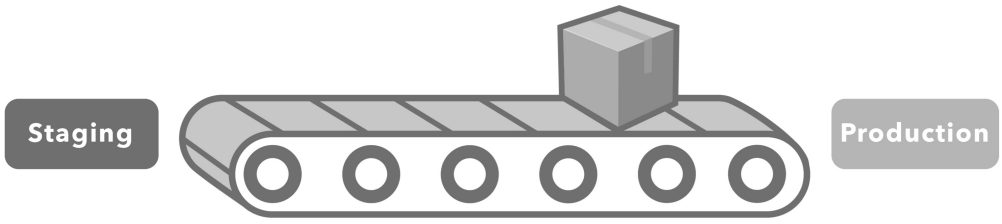
- Automates environment setup (and teardown)
- Manages all infrastructure assets and configurations in version control
- Accelerates and de-risks deployment by provisioning standard/gold environments on demand
- Enables infrastructure to always be in a deployable state (just like application code!)



Notes:

Deployment automation

- Automate all steps from code commit to production deployment
- Store all environment and package information in version control
- Test the deployment process itself



Notes:

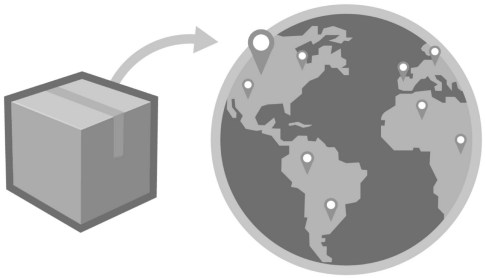
Self Service deployment

- If complete automation from code commit to deployment to production is not possible, automate the deployment of the package to production
- Enable anyone to safely deploy validated packages
- Provide simple controls to facilitate what gets deployed

Notes:

Selective deployment

- Deploy to select production targets or environments
- This can be differentiated by data center, geography, customers
- Enables more flexible and sophisticated release strategies



Notes:

Self-assessment: Deploy sub-dimension

1. Sit — Features are deployed to production every 3+ months; deployments are manual and painful; 'deployed' means 'released'.

2. Crawl — Features are deployed to production at PI boundaries; deployments are mostly manual; 'deployed' means 'released'.

3. Walk — Features are deployed to production every iteration; deployments are mostly automated; some features can be deployed without being released.

4. Run — Features are deployed to production every iteration and fully automated through the continuous delivery pipeline; dark releases are common.

5. Fly — Features are deployed continuously throughout each iteration; Dev teams initiate deployments directly via continuous delivery pipeline tools; release is completely decoupled from deployment; dark releases are the norm.

Place a dot on your DevOps Health Radar

5-13

Notes:

5.2 Verify the Solution

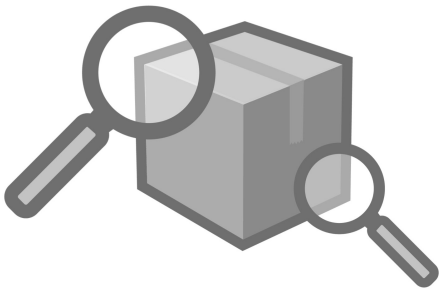
Notes:

Verify

Deploy | **Verify** | Monitor | Respond

Purpose: Assure that deployment solutions behave as expected in production before they are released to end users

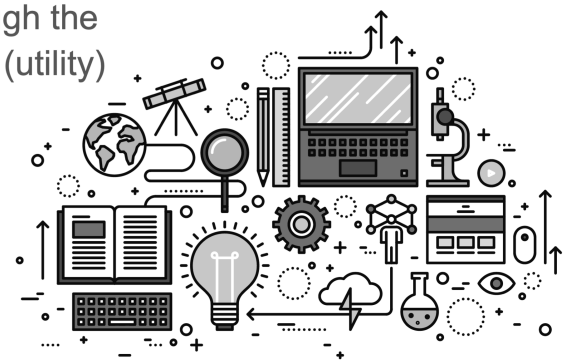
Skills:
- Production testing
- Test automation (covered in build)
- Test data management (covered in Test end-to-end)
- Nonfunctional requirement (covered in Test end-to-end)



Notes:

Production Testing

- Testing of features in the live environment
- This includes functional and nonfunctional testing
- Running synthetic transactions through the services to verify fitness for purpose (utility) and fitness for use (warranty)



5-16

Notes:

Self-assessment: Verify sub-dimension

1. Sit — Deployments are not verified in production before being released to end users.

2. Crawl — Deployments are verified with manual smoke tests and/or UAT; we address deployment issues within a stated grace/warranty period; recovery is painful or impossible; we often correct issues directly in production.

3. Walk — Deployments are verified with manual tests prior to releasing to end users; recovery is painful; we do not make changes directly in production.

4. Run — Deployments are verified using automated production tests, synthetic transactions and security tests prior to release; we can easily roll back or fix forward to recover from failed deployments.

5. Fly — Automated production tests run on an ongoing basis and feed monitoring systems; failed deployments can be rolled back instantly or fixed forward through the entire continuous delivery pipeline.

Place a dot on your DevOps Health Radar

1 min

5-17

Notes:

5.3 Monitor for problems

Notes:

Monitor

Deploy | Verify | **Monitor** | Respond

Purpose: Quantitatively measure system and user behavior in real time

Skills:
- Full-stack telemetry
- Visual displays
- Federated monitoring
- AIOps

Notes:

Full-stack telemetry

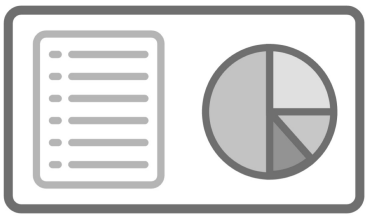
We need proper data to monitor activities:

- Applications should clearly log and report meaningful activities and events
- Architect applications and infrastructure to support telemetry
- Monitor both technical data and business data

Notes:

Visual displays

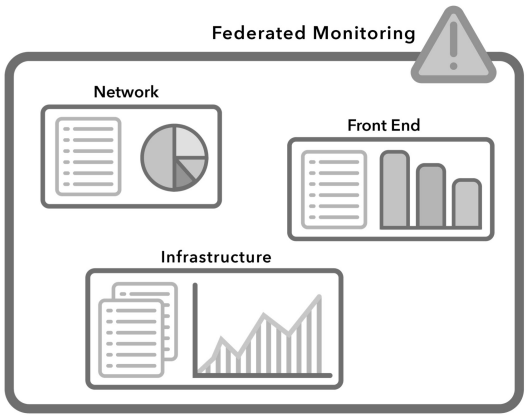
- Visualize telemetry to the entire organization
- Big visible information radiators should project the health of the applications and systems at all times
- Information about key DevOps metrics should also be visible
(time since last deploy, time since last outage, average lead time, etc.)



Notes:

Federated monitoring

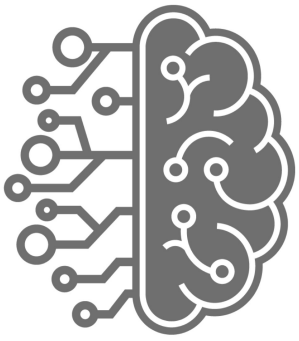
- Aggregate data from various sources into a collection point
- Build big visual information radiators to display the aggregated data
- Provide accessibility and ways to drill down into individual application and infrastructure telemetry



Notes:

Artificial Intelligence for Operations (AIOps)

- The monitoring required for effective DevOps produces a flurry of data, events, and alerts
- AIOps uses machine learning and Big Data techniques to quickly:
  - Aggregate, correlate, and analyze events
  - Separate meaningful events from the “noise”
  - Identify (and predict) root causes of issues
  - Significantly reduce MTTR



Notes:

Self-assessment: Monitor sub-dimension

1. Sit — No feature level production monitoring exists; only infrastructure monitoring is in place.

2. Crawl — Features only log faults and exceptions; analyzing events involves manually correlating logs from multiple systems.

3. Walk — Features log faults, user activity and other events; data is analyzed manually to investigate incidents and measure business value of features.

4. Run — Full-stack monitoring is in place; events can be correlated throughout the architecture; data is presented through system-specific dashboards.

5. Fly — Federated monitoring platform provides one-stop access to full-stack insights; data is used to gauge system performance and business value.

Place a dot on your DevOps Health Radar

SCALED AGILE © Scaled Agile, Inc.

5-24

Notes:

5.4 Respond and recover

Notes:

Respond

Deploy | Verify | Monitor | **Respond**

Purpose: Proactively detect and resolve production issues before they cause business disruption

Skills:
- Chaos Engineering
- Proactive detection
- Cross-team collaboration
- Session replay
- Rollback and fix forward
- Immutable infrastructure
- Version control (covered in Build)

Notes:

Discussion: Chaos Engineering

Step 1: Read the blog post in your Workbook

Step 2: Discuss as a team:

- How did Netflix prepare for the AWS outage?
- Would your systems be prepared for a similar event?
- How might your organization apply chaos engineering?

SCALED AGILE © Scaled Agile, Inc.

5-27

See blog post on the next page >>>

The Netflix Simian Army

Source: Netflix Technology Blog, Jul 19, 2011

The cloud is all about redundancy and fault-tolerance. Since no single component can guarantee 100% uptime (and even the most expensive hardware eventually fails), we have to design a cloud architecture where individual components can fail without affecting the availability of the entire system. In effect, we have to be stronger than our weakest link. We can use techniques like graceful degradation on dependency failures, as well as node-, rack-, datacenter-/availability-zone-, and even regionally-redundant deployments. But just designing a fault tolerant architecture is not enough. We have to constantly test our ability to actually survive these "once in a blue moon" failures.

Imagine getting a flat tire. Even if you have a spare tire in your trunk, do you know if it is inflated? Do you have the tools to change it? And, most importantly, do you remember how to do it right? One way to make sure you can deal with a flat tire on the freeway, in the rain, in the middle of the night is to poke a hole in your tire once a week in your driveway on a Sunday afternoon and go through the drill of replacing it. This is expensive and time-consuming in the real world, but can be (almost) free and automated in the cloud.

This was our philosophy when we built Chaos Monkey, a tool that randomly disables our production instances to make sure we can survive this common type of failure without any customer impact. The name comes from the idea of unleashing a wild monkey with a weapon in your data center (or cloud region) to randomly shoot down instances and chew through cables—all the while we continue serving our customers without interruption.

By running Chaos Monkey in the middle of a business day, in a carefully monitored environment with engineers standing by to address any problems, we can still learn the lessons about the weaknesses of our system, and build automatic recovery mechanisms to deal with them. So next time an instance fails at 3 am on a Sunday, we won't even notice.

Inspired by the success of the Chaos Monkey, we've started creating new simians that induce various kinds of failures, or detect abnormal conditions, and test our ability to survive them; a virtual Simian Army to keep our cloud safe, secure, and highly available.

Latency Monkey induces artificial delays in our RESTful client-server communication layer to simulate service degradation and measures if upstream services respond appropriately. In addition, by making very large delays, we can simulate a node or even an entire service downtime (and test our ability to survive it) without physically bringing these instances down. This can be particularly useful when testing the fault-tolerance of a new service by simulating the failure of its dependencies, without making these dependencies unavailable to the rest of the system.

Conformity Monkey finds instances that don't adhere to best-practices and shuts them down. For example, we know that if we find instances that don't belong to an auto-scaling group, that's trouble waiting to happen. We shut them down to give the service owner the opportunity to re-launch them properly.

Doctor Monkey taps into health checks that run on each instance as well as monitors other external signs of health (e.g. CPU load) to detect unhealthy instances. Once unhealthy instances are detected, they are removed from service and after giving the service owners time to root-cause the problem, are eventually terminated.

Janitor Monkey ensures that our cloud environment is running free of clutter and waste. It searches for unused resources and disposes of them.

Security Monkey is an extension of Conformity Monkey. It finds security violations or vulnerabilities, such as improperly configured AWS security groups, and terminates the offending instances. It also ensures that all our SSL and DRM certificates are valid and are not coming up for renewal.

10–18 Monkey (short for Localization-Internationalization, or l10n-i18n) detects configuration and run time problems in instances serving customers in multiple geographic regions, using different languages and character sets.

Chaos Gorilla is similar to Chaos Monkey, but simulates an outage of an entire Amazon availability zone. We want to verify that our services automatically re-balance to the functional availability zones without user-visible impact or manual intervention.

With the ever-growing Netflix Simian Army by our side, constantly testing our resilience to all sorts of failures, we feel much more confident about our ability to deal with the inevitable failures that we'll encounter in production and to minimize or eliminate their impact to our subscribers. The cloud model is quite new for us (and the rest of the industry); fault-tolerance is a work in progress and we have ways to go to fully realize its benefits. Parts of the Simian Army have already been built, but much remains an aspiration—waiting for talented engineers to join the effort and make it a reality.

Ideas for new simians are coming in faster than we can keep up and if you have ideas, we'd love to hear them! The Simian Army is one of many initiatives we've launched to put the spotlight on increasing the reliability of our service and delivering to our customers an uninterrupted stream of entertainment.

Yury Izrailevsky, Director of Cloud & Systems Infrastructure

Ariel Tseitlin, Director of Cloud Solutions

Proactive detection

- Decoupling deployment from release allows problem detection before problems are exposed to customers
- Proactively look for problems, and practice disaster and recovery situations
- Self sabotage, like chaos monkey, helps build resilience
- These should be coordinated practices in high-assurance environments, along with failure modes and effect analysis (FMEA) during architecture and design

Notes:

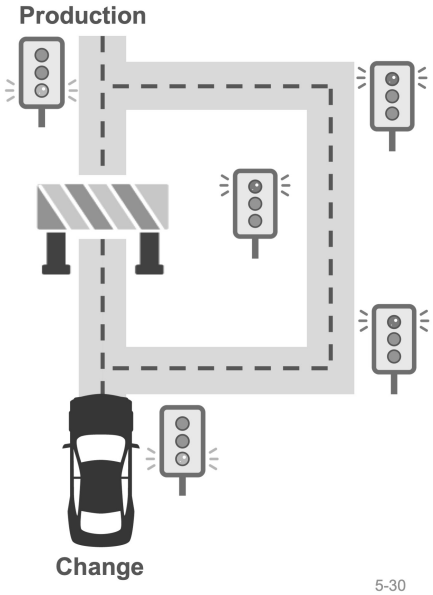
Cross-team collaboration

- Dealing with production issues is everyone's responsibility
- Having a team that can develop and support is preferred
- Teams from across the value stream should collaborate on solving production issues and identifying root causes

Notes:

Immutable infrastructure

- Making changes directly in the production environment creates configuration drifts and has inherent risks
- In an immutable infrastructure environment all changes are deployed through the Continuous Delivery Pipeline.



Make no changes directly in production

Notes:

Session replay

- Record customer sessions and replay them to test problems
- Make session replay available in production, testing, and development environments
- Consider security and privacy when implementing this capability

Notes:

Self-assessment: Respond sub-dimension

1. Sit	—	Customers find issues before we do; resolving high priority issues is time consuming and reactive; customers have low confidence in our ability to recover from production issues.
2. Crawl	—	Operations owns production issues; development involvement requires significant escalation; teams blame each other in times of crisis.
3. Walk	—	Development and Operations collectively own the incident resolution process; recovering from major incidents is reactive but a team effort.
4. Run	—	Our monitoring systems detect most issues before our customers do; Dev and Ops work proactively to recover from major incidents.
5. Fly	—	Our monitoring systems alert us to dangerous conditions based on carefully-designed tolerance thresholds; Developers are responsible for supporting their own code and proactively issue fixes through the pipeline before users are affected.

Place a dot on your DevOps Health Radar

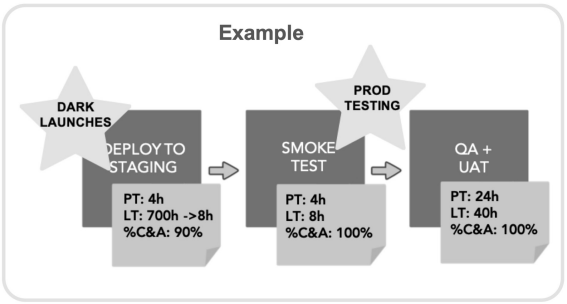
SCALED AGILE © Scaled Agile, Inc.

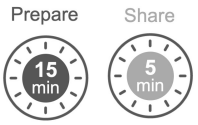
5-32

Notes:

Activity: Apply Continuous Deployment to your Value Stream

- Step 1: Identify the skills in this lesson that would improve flow in your value stream between ready to deploy and ready to release. Write each on a star-shaped sticky note.
- Step 2: Construct a new view of this part of the value stream in the Future State portion of the DevOps Transformation Canvas. (Reuse, reorder, or replace steps as desired.)
- Step 3: Place the stars on the steps they would enable and provide new Process Time, Lead Time, and %C&A for each step.





5-33

Thought organizer

Lesson review

In this lesson you:

- Explored how to deploy to production
- Discussed how to verify the solution
- Analyzed how to monitor for problems
- Discussed how to respond and recover

5-34

Notes:

Lesson 5

Reducing Time-To-Market with Continuous Deployment

Key Learnings & Insights

Lesson 6

Delivering Business Value with Release on Demand

Learning objectives:

6.1 Release on demand

6.2 Stabilize the solution

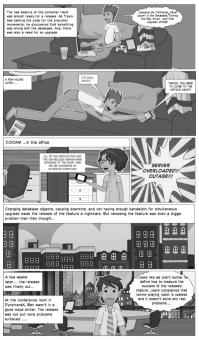
6.3 Measure the business value

6.4 Learn and react

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

Chapter 5: What did we release?

- **Step 1:** Read 'Chapter 5: What did we release?'
- **Step 2:** Review the clues provided by the characters.
- **Step 3:** Considering your own context, discuss:
  - What are some actions you could take to prevent situations like the one at DynotransX from occurring so often?



Prepare 10 min

Share 5 min

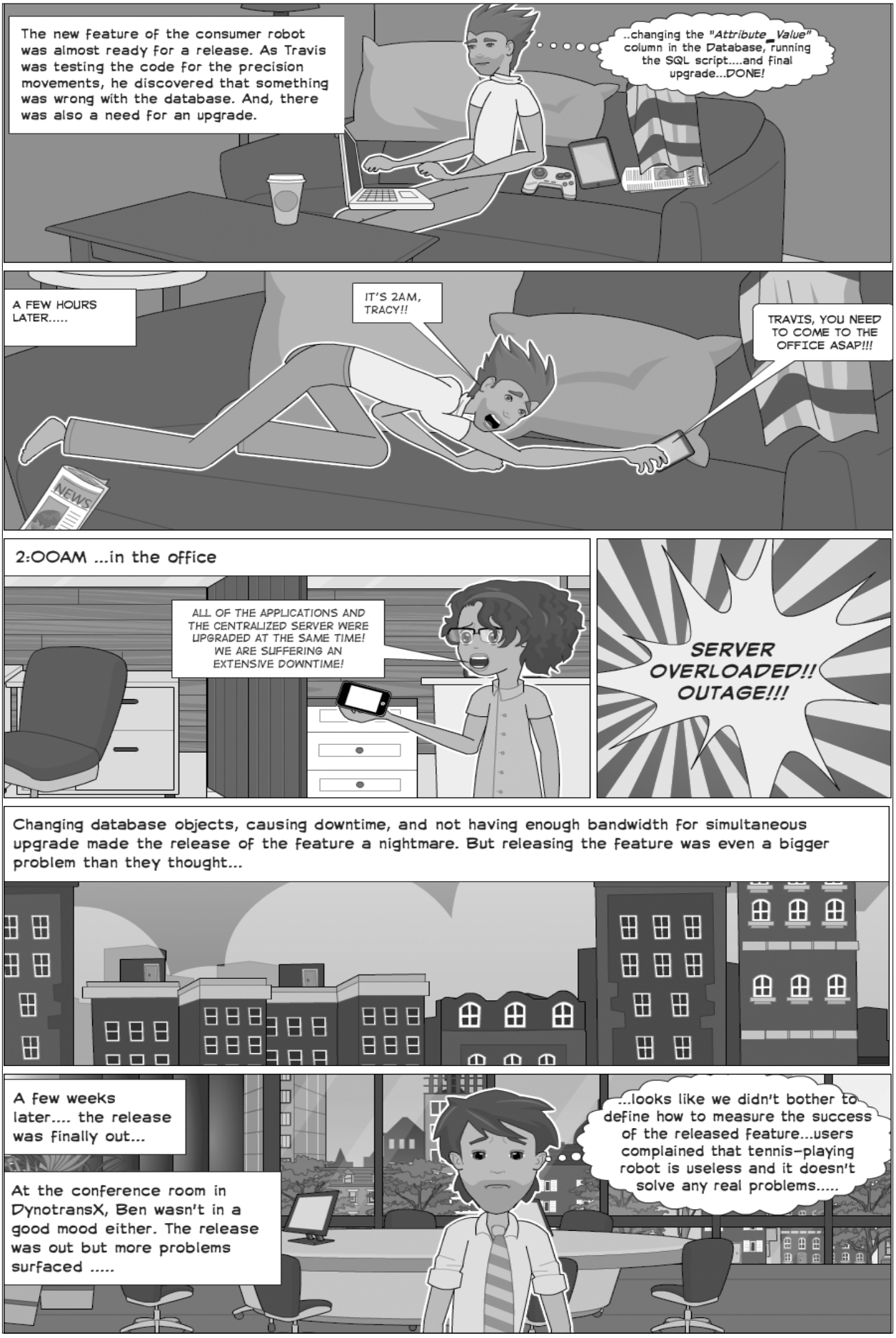
6-3

What could you do in your company to prevent situations like the one at DynotransX from occurring so often?

In your group, share what is your disaster recovery plan.

- ☐ Create a plan for disaster recovery
- ☐ Tightly couple the system architecture to avoid production instability

- ☐ Define clear measurements for success
- ☐ Instead of measuring system performance, fix the technical debt

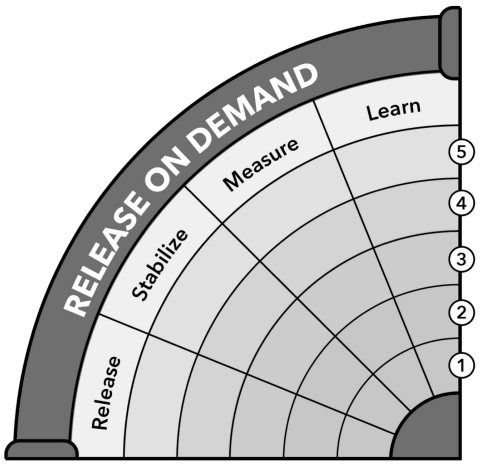

The new feature of the consumer robot was almost ready for a release. As Travis was testing the code for the precision movements, he discovered that something was wrong with the database. And, there was also a need for an upgrade.
..changing the "Attribute_Value" column in the Database, running the SQL script....and final upgrade...DONE!
A FEW HOURS LATER.....
IT'S 2AM, TRACY!!
TRAVIS, YOU NEED TO COME TO THE OFFICE ASAP!!!
NEWS
2:00AM ...in the office
ALL OF THE APPLICATIONS AND THE CENTRALIZED SERVER WERE UPGRADED AT THE SAME TIME! WE ARE SUFFERING AN EXTENSIVE DOWNTIME!
SERVER OVERLOADED!! OUTAGE!!!
Changing database objects, causing downtime, and not having enough bandwidth for simultaneous upgrade made the release of the feature a nightmare. But releasing the feature was even a bigger problem than they thought...
A few weeks later.... the release was finally out...
...looks like we didn't bother to define how to measure the success of the released feature...users complained that tennis-playing robot is useless and it doesn't solve any real problems.....
At the conference room in DynotransX, Ben wasn't in a good mood either. The release was out but more problems surfaced .....

Release dimension

Business objective (The 'Why')
- Business value

IT objective (The 'How')
- Release on Demand

Sub-dimensions (The 'What')
- Release
- Stabilize
- Measure
- Learn



6-4

Notes:

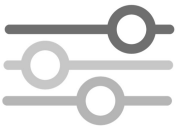
6.1 Release on demand

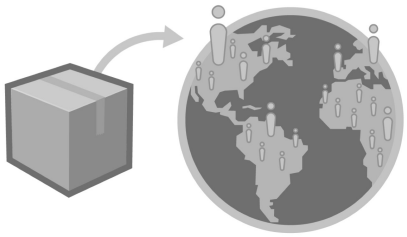
Release

Purpose: Release value to customers all at once or incrementally

Skills:

- Feature toggles (covered in Deploy)
- Canary releases
- Decouple release elements (covered in Architect)
- Dark launches (covered in Deploy)

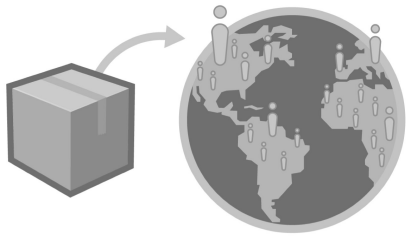




Notes:

Canary releases

- Provide the ability to release value to part of the user population, be it internal or external
- Add or remove user segments based on business decisions
- Combine with selective deployments to enable incremental deployment and rollout



6-7

Notes:

Self-assessment: Release sub-dimension

1. Sit	—	Releases are tightly coupled to deployments and customers are extremely dissatisfied with the frequency of releases.
2. Crawl	—	Releases are tightly coupled to deployments but customers are somewhat dissatisfied with the frequency of releases.
3. Walk	—	Release and deployment are coupled but both occur continuously or on demand.
4. Run	—	Release is decoupled from deployment; deployed features are released to the end user population based on business readiness.
5. Fly	—	Deployed features can be released to individual segments of the user population; feature toggles are refactored when no longer used.

Place a dot on your DevOps Health Radar

1 min

6-8

Notes:

6.2 Stabilize the Solution

Notes:

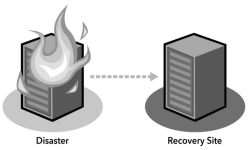
Stabilize

Release | **Stabilize** | Measure | Learn

Purpose: Assure sustainably high levels of business continuity, application service levels, and data protection

Skills:

- Site Reliability Engineering (SRE)
- Failover/disaster recovery
- Continuous security monitoring
- Architect for operations (covered in Architect)
- Nonfunctional requirements (covered in Test End-To-End)



Notes:

Site Reliability Engineering (SRE)

Ensure large systems are highly reliable and scalable via:

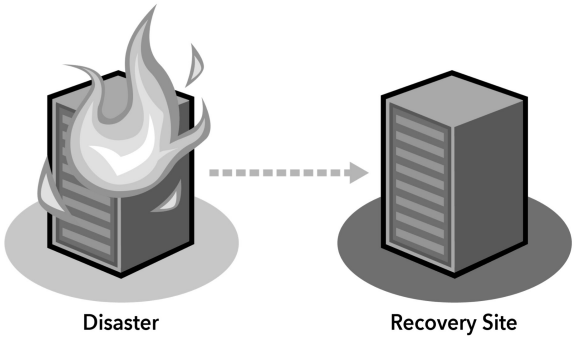
- Shared ownership of system stability between Dev and Ops
- T-shaped engineers with deep development and operations expertise
- Approaching all operational activities as a software concern
- Closely managing Service Level Indicators (SLIs) and Service Level Objectives (SLOs)

Notes:

Failover/disaster recovery

Failures will occur. Develop the ability to recover quickly.

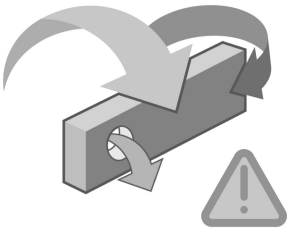
- Failover mechanism allows service to resume quickly, or even avoid service interruption
- Disaster recovery must be planned, architected into the service and practiced



Notes:

Continuous security monitoring

- Detect intrusions and attacks on production services and infrastructure (detective controls)
- Security as code and penetration testing focus on preventing known vulnerabilities from getting to production (preventive controls)
- Test services continuously for newly discovered and reported vulnerabilities



Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring

6-13

Notes:

Self-assessment: Stabilize sub-dimension

1. **Sit** — We experience frequent unplanned outages and/or security breaches with long recovery times.
2. **Crawl** — We experience occasional unplanned outages, but recover within our service level agreements.
3. **Walk** — We have very few unplanned outages; availability, security, and disaster recovery measures are effective.
4. **Run** — We have no unplanned outages; We plan and rehearse failure and recovery.
5. **Fly** — We maximize resiliency by deliberately injecting faults into our production environment and rehearsing recovery procedures.

Place a dot on your DevOps Health Radar

1 min

6-14

Notes:

6.3 Measure the business value

Notes:

Measure

Release | Stabilize | **Measure** | Learn

Purpose: Test hypothesis of business value, and customers are delighted with delivered solution before Operations takes custody

Skills:

- Innovation accounting (covered in Hypothesize)
- Evaluate hypothesis

Notes:

Evaluate hypothesis

- Application telemetry creates a way to evaluate the business results of a hypothesis
- Measure both leading and lagging indicators
- Build the ability to identify the correlation between business results and the hypothesis being tested

Notes:

Self-assessment: Measure sub-dimension

1. Sit	—	We don't define or measure the value of features.
2. Crawl	—	We've defined what 'value' is but don't know how to measure it.
3. Walk	—	We capture qualitative feedback from the business about the value of our features.
4. Run	—	We capture qualitative and quantitative feedback from the business and our monitoring systems about the value of our features.
5. Fly	—	We aggregate the quantitative and qualitative feedback to objectively validate the original hypothesis and inform pivot-or-persevere decisions.

Place a dot on your DevOps Health Radar

1 min

Notes:

6.4 Learn and react

SCALED AGILE © Scaled Agile, Inc. 6-19

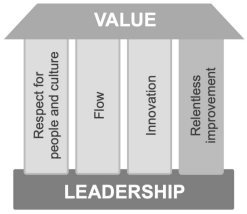
Notes:

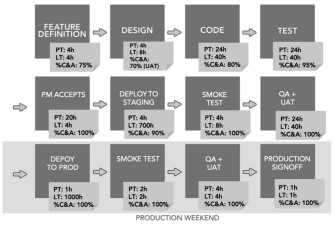
Learn

Release | Stabilize | Measure | **Learn**

Purpose: Learn from the hypothesis on whether to pivot or persevere, as well as how to improve the flow of value

Skills:
- Lean startup (covered in Hypothesize)
- Relentless Improvement
- Value stream mapping



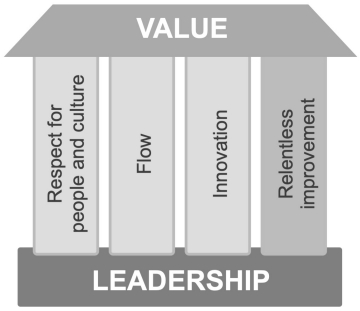


SCALED AGILE © Scaled Agile, Inc. 6-20

Notes:

Relentless improvement

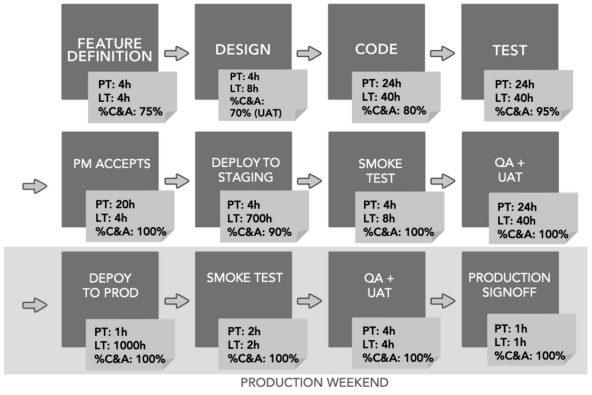
- In order to improve the ability to test hypotheses, the Continuous Delivery Pipeline must be constantly maintained
- Team-level retrospectives and program and solution-level Inspect and Adapt events are crucial to improve the flow
- Focus on the root cause of bottlenecks and research incidents to identify the most important place to improve



Notes:

Value stream mapping

- Continuously apply Value stream mapping to optimize value flow
- Apply current state mapping and future state mapping so objectives are clear



Notes:

Self-assessment: Learn sub-dimension

1. **Sit** — Features are never evaluated post-release.
2. **Crawl** — Features are sometimes evaluated using subjective information.
3. **Walk** — Hypotheses are evaluated using objective measures, but actions are heavily influenced by corporate politics.
4. **Run** — Hypotheses are objectively evaluated; pivot or persevere decisions are made without mercy or guilt.
5. **Fly** — Continuous learning and experimentation are ingrained in the DNA of the organization.

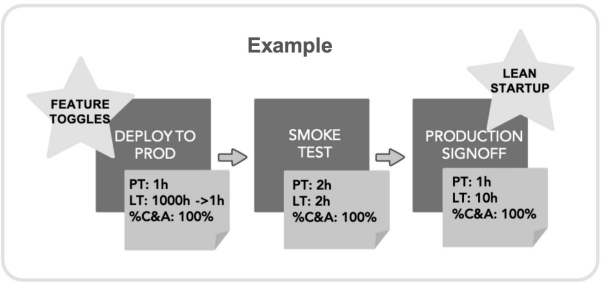
Place a dot on your DevOps Health Radar

6-23

Notes:

Activity: Apply Release on Demand to your Value Stream

- **Step 1:** Identify the skills in this lesson that would improve your value stream between ready to release and hypothesis tested. Write each on a star-shaped sticky note.
- **Step 2:** Construct a new view of this part of the value stream in the Future State portion of the DevOps Transformation Canvas. (Reuse, reorder, or replace steps as desired.)
- **Step 3:** Place the stars on the steps they would enable and provide new Process Time, Lead Time, and %C&A for each step.



Prepare 15 min | Share 5 min

6-24

Thought organizer

Lesson review

In this lesson you:

- Explored how to release on demand
- Discussed how to stabilize the solution
- Explored how to measure the business value
- Discussed how to evaluate a hypothesis and make a decision

Notes:

Lesson 6

Delivering Business Value with Release on Demand

Key Learnings & Insights

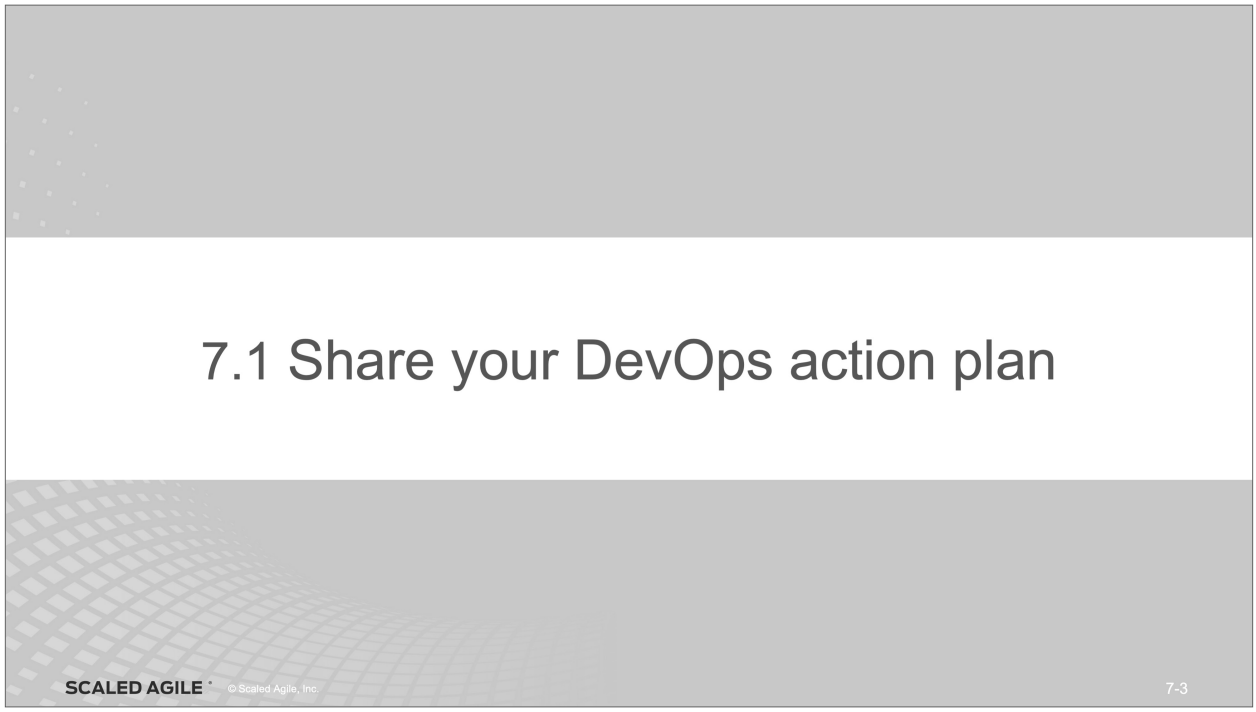
Lesson 7

Taking Action

Learning objectives:

7.1 Share your DevOps action plan

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.


7.1 Share your DevOps action plan
SCALED AGILE
© Scaled Agile, Inc.
7-3

Notes:

Taking Action: Prioritize your DevOps backlog (Part 1)

- **Step 1:** Write down all the improvement items from your board in your workbook.
- **Step 2:** Identify items that are outside the scope of things you can change and document them in the 'boundaries and limitations' section of the DevOps transformation canvas.
- **Step 3:** Identify the item that offers the least overall improvement based on the changes to lead time, process time, or percent complete and accurate. Mark that as '1' in the cost of delay column.

Prepare

7-4

Improvement Item	Cost of delay	Job size	WSJF
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	
	÷	=	

Taking Action: Prioritize your DevOps backlog (Part 2)

- **Step 1:** Estimate the cost of delay for all other items relative to the 1, using the numbers 1,2,3,5,8,13,20
- **Step 2:** Estimate the job size of the items in a similar fashion; smallest job size again being a '1'
- **Step 3**: Calculate WSJF
- **Step 4:** Document the three highest priorities in the 'improvement items' section of the DevOps transformation canvas

Prepare

7-5

Thought organizer

Taking Action: DevOps Transformation Canvas

- **Step 1:** Review other teams' DevOps transformation canvases

> NOTE: Each team should have a person by their canvas to answer questions. Make sure the people who present the canvases rotate and get the chance to walk the gallery.

- **Step 2:** Share some observations with the class

Prepare

7-6

Thought organizer

Chapter 6: The end

- **Step 1:** Read Chapter 6 'The end.'
- **Step 2:** Discuss as a team:
  - What are some of the future challenges you foresee for you and your team while on the DevOps transformation journey?
  - What are some of your takeaways from the story about Travis and Tracy, and the DynotransX, Inc. transformation?

Prepare 8 min

Share 2 min

7-7

Future challenges:

Key takeaways:


ANOTHER LATE CALL
WHAT, TRACY?! ONE OF THE SEVERS CRASHED?!
WHO IS RESPONSIBLE FOR THE SERVERS AND THE DEVICE MANAGEMENT?!
TRAVIS, A SERVER CRASHED AGAIN!!
Tracy has been discussing server issues with Travis and his team ever since the release of the latest feature. It turns out that the servers have been overloading for quite some time.
WE NEED TO ADD ANOTHER PHYSICAL SERVER. THE FIREWALL NEEDS TO BE CHANGED TOO!!
WE ARE GOING TO NEED OS SET UP, SOFTWARE PACKAGES, LICENSE KEYS, ACCOUNTS, IP ADDRESSES, ROUTER PORTS, CABLING.... AND THE FIREWALL?! LAST TIME IT TOOK MONTHS MAKING THE CHANGES TO THE FIREWALL.
WAIT, TRACY!! WHAT DO YOU MEAN? FIREWALL CHANGES SHOULD ONLY TAKE A FEW MINUTES. WHY IS TAKING MONTHS OF WORK?!?!?!
TRAVIS, WE NEED TO SOLVE THIS PROBLEM. WE NEED TO FIND A WAY TO PREVENT DISASTERS LIKE THAT EVER HAPPENING AGAIN! LET'S GET THE TEAMS TOGETHER!!
I AGREE, TRACY! WE NEED TO FIND OUT WHY IT TAKES SO LONG TO CHANGE THE FIREWALL.
BUILD
ACCEPTANCE TEST
TRACY, WHO DOES THE FIREWALL CHANGES?
I WILL FIND OUT
WHO DOES ACCEPTANCE TESTING AND KNOWS HOW LONG IT TAKES?
...finally we are all working together to improve our value stream..


One year later, DynotransX was becoming more and more successful. However, the work wasn't quite done yet. With the plans for scaling, growing and exploring global markets, new challenges were presented to the teams. There was still the need to fully adopt practices for defining the Minimum Viable Product in the context of the Lean Startup Cycle. The Backlog needed preparation, and work had to be prioritized. But this is the nature of challenges – they never stop. The key is to continue finding the best solutions possible...
Tracy no longer had to worry about the downtime of servers and endless support tickets...
....and Travis no longer received 2AM phone calls from Tracy...

Take action after the class

- Share the DevOps transformation canvas and your insights with your peers
- Refine the Value Stream Map with your stakeholders
- Enter your top three improvements into your Program Backlog
- Enhance your software engineering practices with Agile Software Engineering
- Explore the Architect role with SAFe for Architects

7-8

Notes:

Lesson review

In this lesson you:

- Shared your DevOps action plan

Notes:

Lesson 7

Taking Action

Key Learnings & Insights

Lesson 8

Becoming a SAFe DevOps Practitioner

SAFe® Authorized Course Attending this course gives students access to the SAFe DevOps Practitioner exam and related preparation materials.

Video: Become a Certified SAFe Professional

Continue to build on the foundation of SAFe learning you began in class by studying and taking the certification exam.

Earning this certification demonstrates and establishes your new knowledge.

Certification details at:
www.scaledagile.com/certification-and-exam-information-sdp/

https://vimeo.com/307578726

3 min

SCALED AGILE © Scaled Agile, Inc.

8-3

https://vimeo.com/307578726

Scan me

Video: Welcome to the SAFe Community Platform

Want to learn more about the next steps on your SAFe Journey?

Access the SAFe Community Platform and discover all the SAFe resources available for your use!

https://vimeo.com/201877314

5 min

8-4

https://vimeo.com/201877314

Scan me

Lesson 8

Becoming a SAFe DevOps Practitioner

Key Learnings & Insights

SAFe Glossary

Visit the Scaled Agile Framework site
(www.scaledagileframework.com)
to download glossaries translated
into other languages

Guide to acronyms and abbreviations

ART	Agile Release Train
BO	Business Owner
BV	Business Value
BVIR	Big Visual Information Radiator
CapEx	Capital Expenses
CD	Continuous Deployment
CE	Continuous Exploration
CI	Continuous Integration
CFD	Cumulative Flow Diagram
CoD	Cost of Delay
CoP	Community of Practice
DoD	Definition of Done
DSU	Daily Stand-up
EA	Enterprise Architect
EO	Epic Owner
FW	Firmware
HW	Hardware
I&A	Inspect and Adapt
IP	Innovation and Planning (iteration)
KPI	Key Performance Indicator
LPM	Lean Portfolio Management
MBSE	Model-Based Systems Engineering
MMF	Minimum Marketable Feature
MVP	Minimum Viable Product
NFR	Non-functional Requirements
OE	Opportunity Enablement
OpEx	Operating Expenses
PDCA	Plan, Do, Check, Adjust
PI	Program Increment
PM	Product Management
PO/PM	Product Owner / Product Manager
PO	Product Owner
ROAM	Resolved, Owned, Accepted, Mitigated
RR	Risk Reduction
RTE	Release Train Engineer
S4T	SAFe® for Teams
SAFe®	Scaled Agile Framework
SA	SAFe® Agilist
SBD	Set-Based Design
SM	Scrum Master
SMART	Specific, Measurable, Achievable, Realistic, Time-bound
SoS	Scrum of Scrums
SP	SAFe® Practitioner
SPC	SAFe® Program Consultant
STE	Solution Train Engineer
SW	Software
UX	User Experience
VS	Value Stream
VSE	Value Stream Engineer
WIP	Work in Process
WSJF	Weighted Shortest Job First
XP	Extreme Programming

Agile Architecture
Agile Architecture is a set of values and practices that support the active evolution of the design and architecture of a system while implementing new system capabilities.

Agile Release Train (ART)
The Agile Release Train (ART) is a long-lived team of Agile teams, which, along with other stakeholders, develops and delivers solutions incrementally, using a series of fixed-length Iterations within a Program Increment (PI) timebox. The ART aligns teams to a common business and technology mission.

Agile Team
The SAFe Agile Team is a cross-functional group of 5 to 10 people who have the ability and authority to define, build, and test some element of Solution value—all in a short Iteration timebox. Specifically, the SAFe Agile Team incorporates the Dev Team, Scrum Master, and Product Owner roles.

Architectural Runway
The Architectural Runway consists of the existing code, components, and technical infrastructure needed to implement near-term features without excessive redesign and delay.

Built-In Quality
Built-In Quality practices ensure that each Solution element, at every increment, meets appropriate quality standards throughout development.

Business Owners
Business Owners are a small group of stakeholders who have the primary business and technical responsibility for governance, compliance, and return on investment (ROI) for a Solution developed by an Agile Release Train (ART). They are key stakeholders on the ART who must evaluate fitness for use and actively participate in certain ART events.

CapEx and OpEx
Capital Expenses (CapEx) and Operating Expenses (OpEx) describe Lean-Agile financial accounting practices in a Value Stream budget. In some cases, CapEx may include capitalized labor associated with the development of intangible assets—such as software, intellectual property, and patents.

Capabilities
A Capability is a higher-level solution behavior that typically spans multiple ARTs. Capabilities are sized and split into multiple features to facilitate their implementation in a single PI.

Communities of Practice (CoPs)
Communities of Practice (CoPs) are organized groups of people who have a common interest in a specific technical or business domain. They collaborate regularly to share information, improve their skills, and actively work on advancing the general knowledge of the domain.

Compliance
Compliance refers to a strategy and a set of activities and artifacts that allow teams to apply Lean-Agile development methods to build systems that have the highest possible quality, while simultaneously assuring they meet any regulatory, industry, or other relevant standards.

Continuous Delivery Pipeline
The Continuous Delivery Pipeline (also referred to as 'pipeline') represents the workflows, activities, and automation needed to provide a continuous release of value to the end user.

Continuous Deployment (CD)
Continuous Deployment (CD) is the process that takes validated Features from Continuous Integration and deploys them into the production environment, where they are tested and readied for release. It is the third element in the four-part Continuous Delivery Pipeline of Continuous Exploration (CE), Continuous Integration (CI), Continuous Deployment, and Release on Demand.

Continuous Exploration (CE)
Continuous Exploration (CE) is the process of continually exploring the market and user needs, and defining a Vision, Roadmap, and set of Features that address those needs. It's the first element in the four-part Continuous Delivery Pipeline, preceding Continuous Integration (CI) Continuous Deployment (CD), and Release on Demand.

Continuous Integration (CI)
Continuous Integration (CI) is the process of taking features from the Program Backlog and developing, testing, integrating, and validating them in a staging environment where they are ready for deployment and release.

Core Values
The four Core Values of alignment, built-in quality, transparency, and program execution represent the fundamental beliefs that are key to SAFe's effectiveness. These guiding principles help dictate behavior and action for everyone who participates in a SAFe portfolio.

Customers
Customers are the ultimate buyer of every Solution. They are an integral part of the Lean-Agile development process and Value Stream and have specific responsibilities in SAFe.

Dev Team
The Dev Team is a subset of the Agile Team. It consists of the dedicated professionals who can develop and test a Story, Feature, or component. The Dev Team typically includes software developers and testers, engineers, and other dedicated specialists required to complete a vertical slice of functionality.

DevOps
DevOps is a mindset, a culture, and a set of technical practices. It provides communication, integration, automation, and close cooperation among all the people needed to plan, develop, test, deploy, release, and maintain a Solution.

Develop on Cadence
Develop on Cadence is an essential method for managing the inherent variability of systems development in a flow-based system, by making sure important events and activities occur on a regular, predictable schedule.

Economic Framework
The Economic Framework is a set of decision rules that align everyone to the financial objectives of the Solution and guides the economic decision-making process. It contains four primary constructs: Lean Budgets, Epic funding and governance, decentralized decision-making, and job sequencing based on the Cost of Delay (CoD).

Enablers
Enablers support the activities needed to extend the Architectural Runway to provide future business functionality. These include exploration, infrastructure, compliance, and architecture development. They are captured in the various backlogs and occur at all levels of the Framework.

Enterprise
The Enterprise represents the business entity to which each SAFe portfolio belongs.

Enterprise Architect
The Enterprise Architect promotes adaptive design, and engineering practices and drives architectural initiatives for the portfolio. Enterprise Architects also facilitate the reuse of ideas, components, services, and proven patterns across various solutions in a portfolio.

Epic
An Epic is a container for a Solution development initiative large enough to require analysis, the definition of a Minimum Viable Product (MVP), and financial approval prior to implementation. Implementation occurs over multiple Program Increments (PIs) and follows the Lean startup 'build-measure-learn' cycle.

Epic Owners
Epic Owners are responsible for coordinating portfolio Epics through the Portfolio Kanban system. They define the epic, its Minimum Viable Product (MVP), and Lean business case, and when approved, facilitate implementation.

Essential SAFe configuration
The Essential SAFe configuration is the heart of the Framework and is the simplest starting point for implementation. It's the basic building block for all other SAFe configurations and describes the most critical elements needed to realize the majority of the Framework's benefits.

Features
A Feature is a service that fulfills a stakeholder need. Each feature includes a benefit hypothesis and acceptance criteria, and is sized or split as necessary to be delivered by a single Agile Release Train (ART) in a Program Increment (PI).

Foundation
The Foundation contains the supporting principles, values, mindset, implementation guidance, and leadership roles needed to deliver value successfully at scale.

Full SAFe configuration
The Full SAFe configuration is the most comprehensive version of the Framework. It supports enterprises that build and maintain large integrated solutions, which require hundreds of people or more, and includes all levels of SAFe: team, program, large solution, and portfolio. In the largest enterprises, multiple instances of various SAFe configurations may be required.

Innovation and Planning Iteration
The Innovation and Planning (IP) Iteration occurs every Program Increment (PI) and serves multiple purposes. It acts as an estimating buffer for meeting PI Objectives and provides dedicated time for innovation, continuing education, PI Planning, and Inspect and Adapt (I&A) events.

Inspect & Adapt (I&A)
The Inspect and Adapt (I&A) is a significant event, held at the end of each Program Increment (PI), where the current state of the Solution is demonstrated and evaluated by the train. Teams then reflect and identify improvement backlog items via a structured, problem-solving workshop.

Iteration
Iterations are the basic building block of Agile development. Each iteration is a standard, fixed-length timebox, where Agile Teams deliver incremental value in the form of working, tested software and systems. The recommended duration of the timebox is two weeks. However, one to four weeks is acceptable, depending on the business context.

Iteration Execution
Iteration Execution is how Agile Teams manage their work throughout the Iteration timebox, resulting in a high-quality, working, tested system increment.

Iteration Goals
Iteration Goals are a high-level summary of the business and technical goals that the Agile Team agrees to accomplish in an Iteration. They are vital to coordinating an Agile Release Train (ART) as a self-organizing, self-managing team of teams.

Iteration Planning
Iteration Planning is an event where all team members determine how much of the Team Backlog they can commit to delivering during an upcoming Iteration. The team summarizes the work as a set of committed Iteration Goals.

Iteration Retrospective
The Iteration Retrospective is a regular meeting where Agile Team members discuss the results of the Iteration, review their practices, and identify ways to improve.

Iteration Review
The Iteration Review is a cadence-based event, where each team inspects the increment at the end of every Iteration to assess progress, and then adjusts its backlog for the next iteration.

Large Solution Level
The Large Solution Level contains the roles, artifacts, and processes needed to build large and complex solutions. This includes a stronger focus on capturing requirements in Solution Intent, the coordination of multiple Agile Release Trains (ARTs) and Suppliers, and the need to ensure compliance with regulations and standards.

Large Solution SAFe configuration
The Large Solution SAFe configuration is for developing the largest and most complex solutions that typically require multiple Agile release trains and Suppliers, but do not require portfolio-level considerations. This is common for industries like aerospace and defense, automotive, and government, where the large solution—not portfolio governance—is the primary concern.

Lean Budgets
Lean Budgets is a set of practices that minimize overhead by funding and empowering Value Streams rather than projects while maintaining financial and fitness-for-use governance. This is achieved through objective evaluation of working systems, active management of Epic investments, and dynamic budget adjustments.

Lean Portfolio Management (LPM)
The Lean Portfolio Management (LPM) function has the highest level of decision-making and financial accountability for the products and Solutions in a SAFe portfolio.

Lean User Experience (Lean UX)
Lean User Experience (Lean UX) design is a mindset, culture, and a process that embraces Lean-Agile methods. It implements functionality in minimum viable increments and determines success by measuring results against a benefit hypothesis.

Lean and Agile Principles
SAFe is based on nine immutable, underlying Lean and Agile Principles. These tenets and economic concepts inspire and inform the roles and practices of SAFe.

Lean-Agile Leaders
Lean-Agile Leaders are lifelong learners who are responsible for the successful adoption of SAFe and the results it delivers. They empower and help teams build better systems by learning, exhibiting, teaching and coaching SAFe's Lean-Agile principles and practices.

Lean-Agile Mindset
The Lean-Agile Mindset is the combination of beliefs, assumptions, and actions of SAFe leaders and practitioners who embrace the concepts of the Agile Manifesto and Lean thinking. It's the personal, intellectual, and leadership foundation for adopting and applying SAFe principles and practices.

Metrics
Metrics are agreed-upon measures used to evaluate how well the organization is progressing toward the portfolio, large solution, program, and team's business and technical objectives.

Milestones
Milestones are used to track progress toward a specific goal or event. There are three types of SAFe milestones: Program Increment (PI), fixed-date, and learning milestones.

Model-Based Systems Engineering (MBSE)
Model-Based Systems Engineering (MBSE) is the practice of developing a set of related system models that help define, design, and document a system under development. These models provide an efficient way to explore, update, and communicate system aspects to stakeholders, while significantly reducing or eliminating dependence on traditional documents.

Nonfunctional Requirements (NFRs)
Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

Portfolio Backlog
The Portfolio Backlog is the highest-level backlog in SAFe. It provides a holding area for upcoming business and enabler Epics intended to create a comprehensive set of Solutions, which provides the competitive differentiation and operational improvements needed to address the Strategic Themes and facilitate business success.

Portfolio Kanban
The Portfolio Kanban is a method used to visualize, manage, and analyze the prioritization and flow of portfolio Epics from ideation to implementation and completion.

Portfolio Level
The Portfolio Level contains the principles, practices, and roles needed to initiate and govern a set of development Value Streams. This is where strategy and investment funding are defined for value streams and their Solutions. This level also provides Agile portfolio operations and Lean governance for the people and resources needed to deliver solutions.

Portfolio SAFe configuration
The Portfolio SAFe configuration helps align portfolio execution to the enterprise strategy, by organizing Agile development around the flow of value, through one or more value streams. It provides business agility through principles and practices for portfolio strategy and investment funding, Agile portfolio operations, and Lean governance.

Pre-and Post-PI Planning
Pre– and Post–Program Increment (PI) Planning events are used to prepare for, and follow up after, PI Planning for Agile Release Trains (ARTs) and Suppliers in a Solution Train.

Product Management
Product Management has content authority for the Program Backlog. They are responsible for identifying Customer needs, prioritizing Features, guiding the work through the Program Kanban and developing the program Vision and Roadmap.

Product Owner (PO)
The Product Owner (PO) is a member of the Agile Team responsible for defining Stories and prioritizing the Team Backlog to streamline the execution of program priorities while maintaining the conceptual and technical integrity of the Features or components for the team.

Program Backlog
The Program Backlog is the holding area for upcoming Features, which are intended to address user needs and deliver business benefits for a single Agile Release Train (ART). It also contains the enabler features necessary to build the Architectural Runway.

Program Increment (PI)
A Program Increment (PI) is a timebox during which an Agile Release Train (ART) delivers incremental value in the form of working, tested software and systems. PIs are typically 8 – 12 weeks long. The most common pattern for a PI is four development Iterations, followed by one Innovation and Planning (IP) Iteration.

Program Increment (PI) Planning
Program Increment (PI) Planning is a cadence-based, face-to-face event that serves as the heartbeat of the Agile Release Train (ART), aligning all the teams on the ART to a shared mission and Vision.

Program Kanban
The Program and Solution Kanban systems are a method to visualize and manage the flow of Features and Capabilities from ideation to analysis, implementation, and release through the Continuous Delivery Pipeline.

Program Level
The Program Level contains the roles and activities needed to continuously deliver solutions via an Agile Release Train (ART).

Refactoring
Refactoring is the activity of improving the internal structure or operation of a code or component without changing its external behavior.

Release Train Engineer (RTE)
The Release Train Engineer (RTE) is a servant leader and coach for the Agile Release Train (ART). The RTE's major responsibilities are to facilitate the ART events and processes and assist the teams in delivering value. RTEs communicate with stakeholders, escalate impediments, help manage risk, and drive relentless improvement.

Release on Demand
Release on Demand is the process by which Features deployed into production are released incrementally or immediately to Customers based on market demand.

Roadmap
The Roadmap is a schedule of events and Milestones that communicate planned Solution deliverables over a timeline. It includes commitments for the planned, upcoming Program Increment (PI) and offers visibility into the deliverables forecasted for the next few PIs.

SAFe Implementation Roadmap
The SAFe Implementation Roadmap consists of an overview graphic and a 12-article series that describes a strategy and an ordered set of activities that have proven to be effective in successfully implementing SAFe.

SAFe Program Consultants (SPCs)
SAFe Program Consultants (SPCs) are change agents who combine their technical knowledge of SAFe with an intrinsic motivation to improve the company's software and systems development processes. They play a critical role in successfully implementing SAFe. SPCs come from numerous internal or external roles, including business and technology leaders, portfolio/program/project managers, process leads, architects, analysts, and consultants.

Scrum Master
Scrum Masters are servant leaders and coaches for an Agile Team. They help educate the team in Scrum, Extreme Programming (XP), Kanban, and SAFe, ensuring that the agreed Agile process is being followed. They also help remove impediments and foster an environment for high-performing team dynamics, continuous flow, and relentless improvement.

ScrumXP
ScrumXP is a lightweight process to deliver value for cross-functional, self-organized teams within SAFe. It combines the power of Scrum project management practices with Extreme Programming (XP) practices.

Set-Based Design
Set-Based Design (SBD) is a practice that keeps requirements and design options flexible for as long as possible during the development process. Instead of choosing a single point solution upfront, SBD identifies and simultaneously explores multiple options, eliminating poorer choices over time. It enhances flexibility in the design process by committing to technical solutions only after validating assumptions, which produces better economic results.

Shared Services
Shared Services represents the specialty roles, people, and services that are necessary for the success of an Agile Release Train (ART) or Solution Train but that cannot be dedicated full-time.

Solution
Each Value Stream produces one or more Solutions, which are products, services, or systems delivered to the Customer, whether internal or external to the Enterprise.

Solution Architect/Engineer
The Solution Architect/Engineering role represents an individual or small team that defines a shared technical and architectural vision for the Solution under development. They participate in determining the system, subsystems, and interfaces, validate technology assumptions and evaluate alternatives, working closely with the Agile Release Train (ARTs) and Solution Train.

Solution Backlog
The Solution Backlog is the holding area for upcoming Capabilities and enablers, each of which can span multiple ARTs and is intended to advance the Solution and build its architectural runway.

Solution Context
Solution Context identifies critical aspects of the operational environment for a Solution. It provides an essential understanding of requirements, usage, installation, operation, and support of the solution itself. Solution context heavily influences opportunities and constraints for releasing on demand.

Solution Demo
The Solution Demo is where the results of development efforts from the Solution Train are integrated, evaluated, and made visible to Customers and other stakeholders.

Solution Management
Solution Management has content authority for the Solution Backlog. They work with customers to understand their needs, prioritize Capabilities, create the Solution vision and roadmap, define requirements, and guide work through the Solution Kanban.

Solution Train
The Solution Train is the organizational construct used to build large and complex Solutions that require the coordination of multiple Agile Release Trains (ARTs), as well as the contributions of Suppliers. It aligns ARTs with a shared business and technology mission using the solution Vision, Backlog, and Roadmap, and an aligned Program Increment (PI)

Spanning Palette
The Spanning Palette contains various roles and artifacts that may be applicable to a specific team, program, large solution, or portfolio context. A key element of SAFe's flexibility and configurability, the spanning palette permits organizations to apply only the elements needed for their configuration.

Spikes
Spikes are a type of exploration Enabler Story in SAFe. Defined initially in Extreme Programming (XP), they represent activities such as research, design, investigation, exploration, and prototyping. Their purpose is to gain the knowledge necessary to reduce the risk of a technical approach, better understand a requirement, or increase the reliability of a story estimate.

Stories
Stories are short descriptions of a small piece of desired functionality, written in the user's language. Agile Teams implement small, vertical slices of system functionality and are sized so they can be completed in a single Iteration.

Supplier
A Supplier is an internal or external organization that develops and delivers components, subsystems, or services that help Solution Trains provide Solutions to their Customers.

System Demo
The System Demo is a significant event that provides an integrated view of new Features for the most recent Iteration delivered by all the teams in the Agile Release Train (ART). Each demo gives ART stakeholders an objective measure of progress during a Program Increment (PI).

System Team
The System Team is a specialized Agile Team that assists in building and using the Agile development environment, including Continuous Integration, test automation, and Continuous Deployment. The System Team supports the integration of assets from Agile teams, performs end-to-end Solution testing where necessary, and assists with deployment and release.

Team Backlog
The Team Backlog contains user and enabler Stories that originate from the Program Backlog, as well as stories that arise locally from the team's local context. It may include other work items as well, representing all the things a team needs to do to advance their portion of the system.

Team Kanban
Team Kanban is a method that helps teams facilitate the flow of value by visualizing workflow, establishing Work In Process (WIP) limits, measuring throughput, and continuously improving their process.

Team Level
The Team Level contains the roles, activities, events, and processes which Agile Teams build and deliver value in the context of the Agile Release Train (ART).

Test-First
Test-First is a Built-In Quality practice derived from Extreme Programming (XP) that recommends building tests before writing code to improve delivery by focusing on the intended results.

Value Stream Coordination
Value Stream Coordination provides guidance to manage dependencies and exploit the opportunities in a portfolio.

Value Streams
Value Streams represent the series of steps that an organization uses to build Solutions that provide a continuous flow of value to a Customer. SAFe value streams are used to define and realize Portfolio-level business objectives and organize Agile Release Trains (ARTs) to deliver value more rapidly.

Vision
The Vision is a description of the future state of the Solution under development. It reflects Customer and stakeholder needs, as well as the Feature and Capabilities, proposed to meet those needs.

Weighted Shortest Job First (WSJF)
Weighted Shortest Job First (WSJF) is a prioritization model used to sequence jobs (ex., Features, Capabilities, and Epics) to produce maximum economic benefit. In SAFe, WSJF is estimated as the Cost of Delay (CoD) divided by job size.

Get SAFe® Certified!

Build alignment across teams and the enterprise

Scaled Agile offers a portfolio of credentials designed to meet the needs of Lean-Agile professionals throughout their career. Each certification is supported by world-class courseware and value-added resources that prepare the individual to succeed as a key player in a SAFe enterprise. **The result is higher-quality implementations, and greater stability for the organization.**

EXECUTIVES, MANAGERS, & STAKEHOLDERS

Leading SAFe®

2 day course

With SAFe® 4 Agilist Certification

Develop a skillset that's in demand worldwide when you become a SAFe 4 Agilist (SA). During this two-day course, you'll learn the principles and practices of the SAFe®, how to execute and release value through Agile Release Trains, and what it means to lead a Lean-Agile transformation at enterprise scale.

AGILE TEAMS

SAFe® For Teams

2 day course

With SAFe® 4 Practitioner Certification

Build the skills needed to become a high-performing team member of an Agile Release Train (ART)—and learn how to collaborate effectively with other teams—by becoming a SAFe® 4 Practitioner (SP). During this two-day course, attendees will gain an in-depth understanding of the ART, how it delivers value, and what they can do to effectively perform their role using Scrum, Kanban, and XP.

LEAN-AGILE CHANGE AGENTS & CONSULTANTS

Implementing SAFe®

4 day course

With SAFe® 4 Program Consultant Certification

Become a leading agent for change—and enable your enterprise to succeed in a disruptive marketplace—when you certify as a SAFe® 4 Program Consultant (SPC). During this four-day course, you'll learn how to lead a Lean-Agile transformation by leveraging the practices and principles of SAFe®. You will learn how to effectively coach programs, launch Agile Release Trains, build a continuous delivery pipeline and DevOps culture, and empower a Lean Portfolio. The first two days of the course—Leading SAFe®—will provide you with the basis to teach SAFe to others.

AGILE RELEASE TRAIN MEMBERS, SAFe DevOps

SAFe® DevOps

2 day course

With SAFe® 4 DevOps Practitioner Certification

This two-day course provides a comprehensive overview of the DevOps competencies needed to accelerate time-to-market by improving the flow of value through the Continuous Delivery Pipeline. Attendees will map the current value stream through their delivery pipeline from idea to cash, and identify practices that will eliminate bottlenecks to flow.

NEW SCRUM MASTERS

SAFe® Scrum Master

2 day course

With SAFe® 4 Scrum Master Certification

Attendees will gain an understanding of the role of Scrum Master in a SAFe enterprise. Unlike traditional Scrum Master training that focuses on the fundamentals of team-level Scrum, the SAFe® Scrum Master course explores the role of Scrum in the context of the entire enterprise, and prepares individuals to successfully plan and execute the Program Increment (PI), the primary enabler of alignment throughout all levels of a SAFe® organization.

PRACTICING SCRUM MASTERS

SAFe® Advanced Scrum Master

2 day course

With SAFe® 4 Advanced Scrum Master Certification

This course prepares current Scrum Masters for their leadership role in facilitating Agile team, program, and enterprise success in a SAFe® implementation. The course covers facilitation of cross-team interactions in support of the program execution and relentless improvement. It enhances the Scrum paradigm with an introduction to scalable engineering and DevOps practices; the application of Kanban to facilitate the flow of value; and supporting interactions with architects, product management, and other critical stakeholders in the larger program and enterprise contexts.

PRODUCT OWNERS, PRODUCT MANAGERS

SAFe® Product Owner/Product Manager

2 day course

With SAFe® 4 Product Owner/Product Manager Certification

In this two-day course, you will learn how the roles of Product Owner, Product Manager, Solution Manager, and Epic Owner drive the delivery of value in the SAFe enterprise. You'll get an overview of the SAFe®, the Lean-Agile mindset, and an understanding of how the PO and PM roles operate in the enterprise to drive the delivery of value. Finally, you will get an in-depth understanding of the specific activities, tools, and mechanics used to effectively deliver value to the enterprise.

RELEASE TRAIN ENGINEERS

SAFe® Release Train Engineer

3 day workshop

SAFe® 4 Release Train Engineer Certification

In this advanced workshop, you will gain an in-depth understanding of the role and responsibilities of a Release Train Engineer (RTE) in the SAFe enterprise. Through experiential learning, you will learn how to facilitate and enable end-to-end value delivery through Agile Release Trains (ARTs) and value streams. You will also learn how to build a high-performing ART by becoming a servant leader and coach, and how to plan and execute a Program Increment (PI) planning event, the primarily enabler of alignment throughout all levels of a SAFe organization.

SAFe® PROVIDED BY SCALED AGILE®

scaledagile.com/calendar