NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Course Agenda

- Module 01: What is Continuous Delivery?
- Module 02: Prerequisites to Continuous Delivery
- Module 03: Implementing Continuous Delivery
- Module 04: Continuous Integration
- Module 05: CD and Your Environments
- Module 06: The Deployment Pipeline
- Module 07: Design of a Strong Foundation

TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Agenda

- Module 08: Automated Testing
- Module 09: Developing on Trunk
- Module 10: Telemetry, Logging & Monitoring
- Module 11: Security in CI/CD
- Module 12: Code Stability
- Module 13: The Planning Paradigm
- Module 14: Setting Goals and Getting Started

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Who is this workshop for?

- Developers and their leaders
- Teams responsible for:
  - Code
  - Infrastructure
  - QA, test and security
  - Operations
- Anyone involved in creating or deploying software solutions – and the outcomes

NOTES:

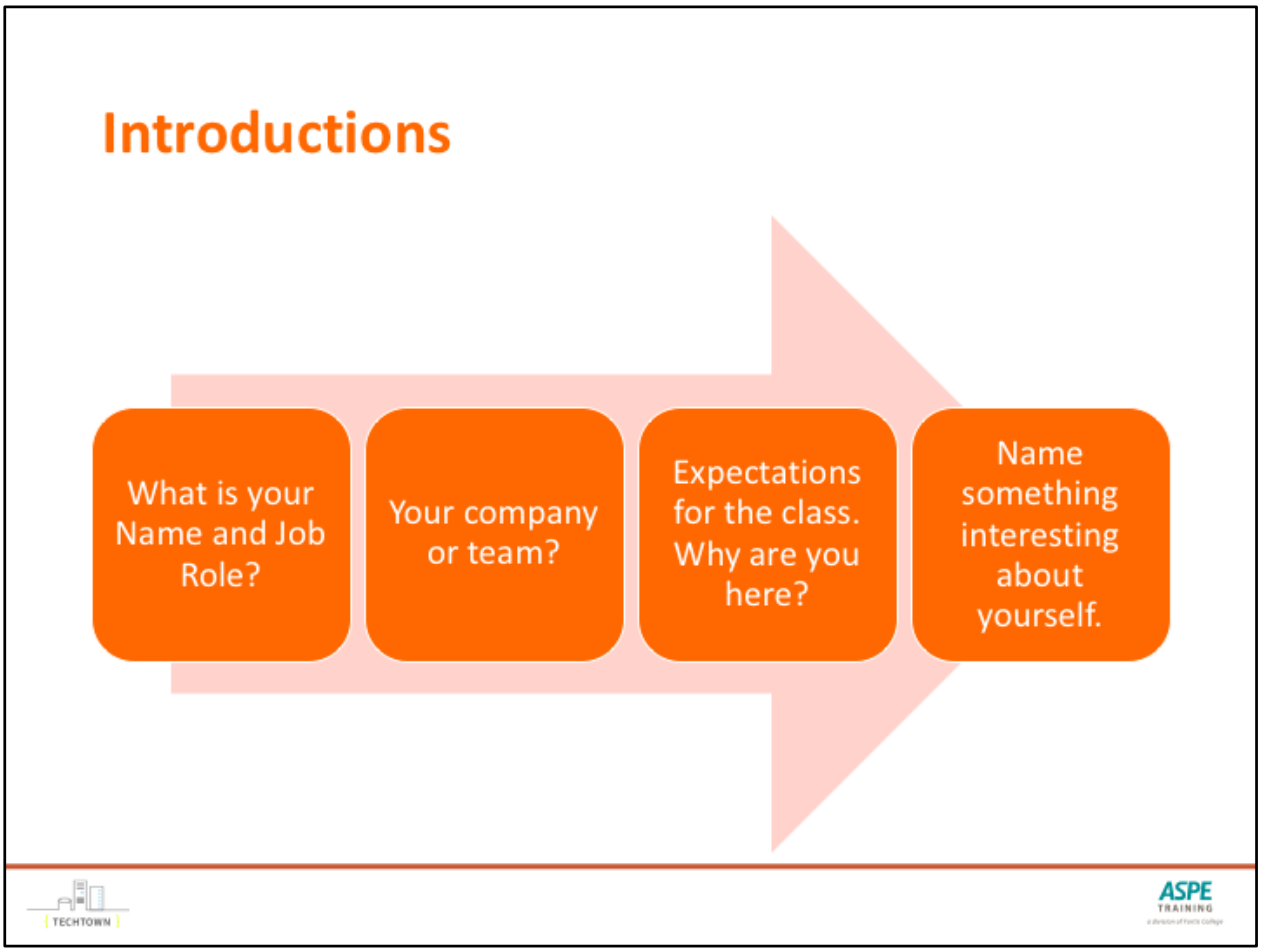
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Related Recourses

- ***Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*** – by Jez Humble and David Farley
- ***Leading the Transformation: Applying Agile and DevOps Principles at Scale*** – by Gary Gruver and Tommy Mouser
- ***The DevOps Handbook*** – by Gene Kim, Jez Humble, Patrick DeBois, John Willis, and John Allspaw
- ***Toyota Kata*** – by Mike Rother

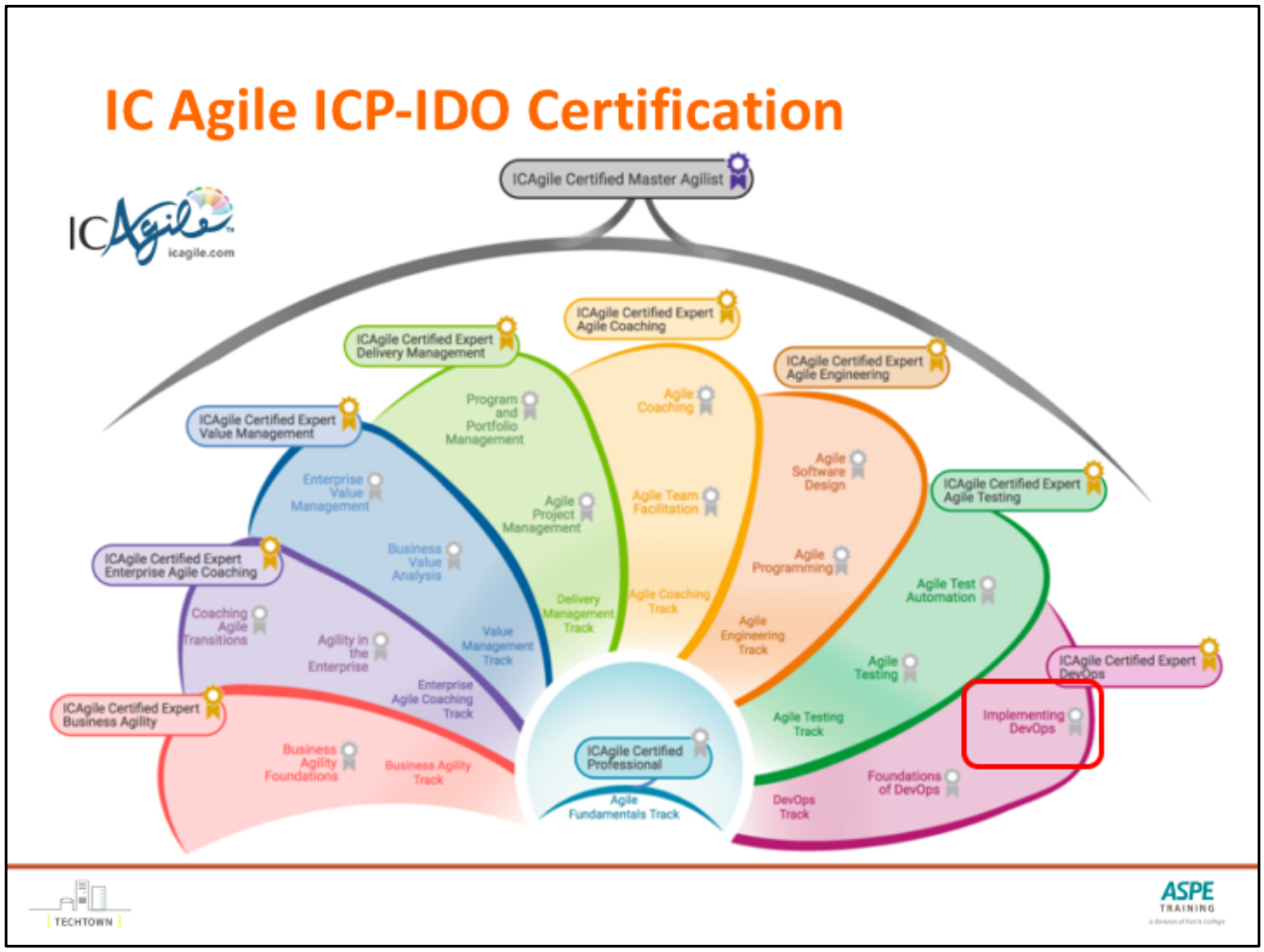
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



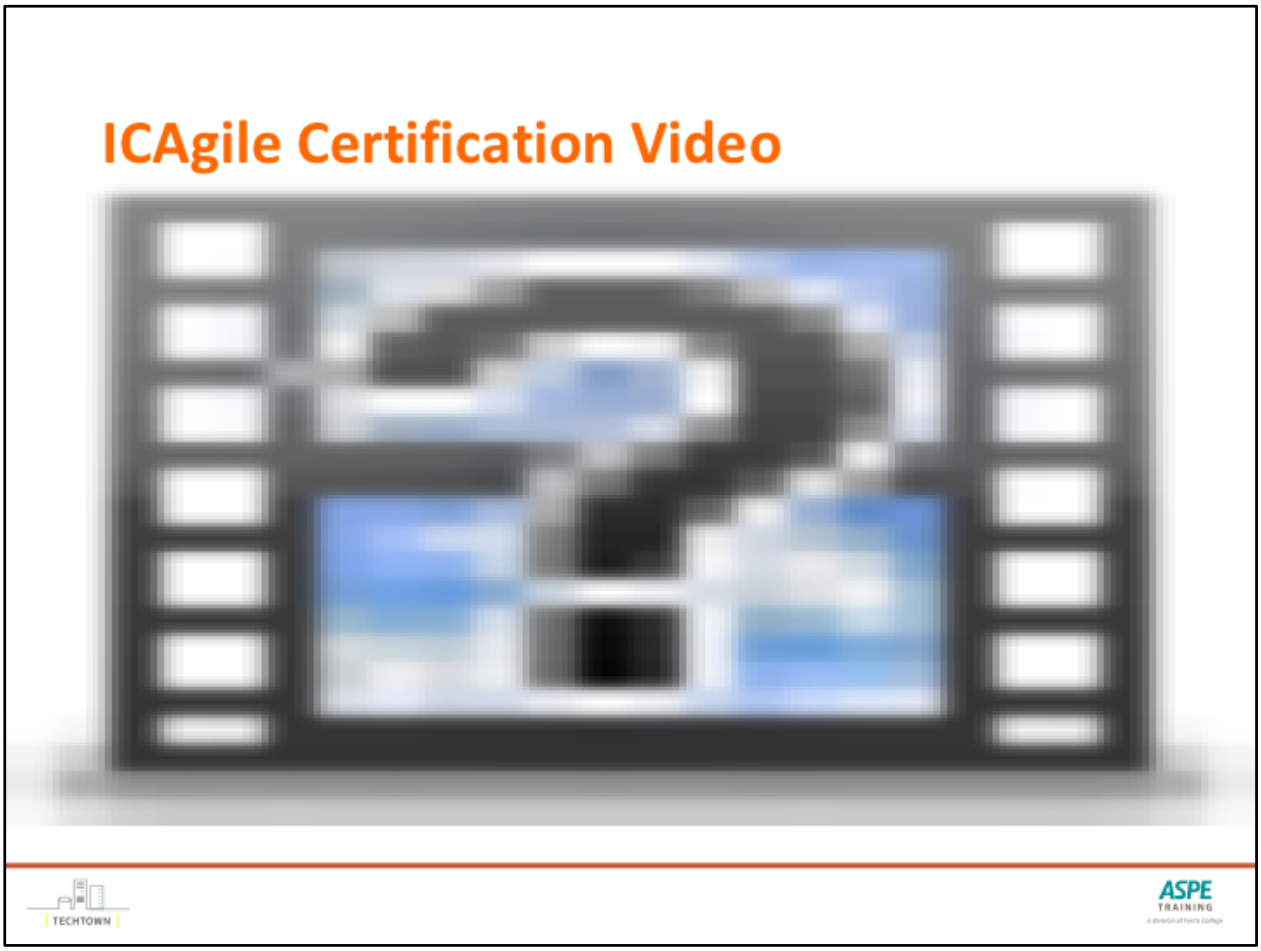
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



For more information about this certification:
http://icagile.com/Learning-Roadmap/DevOps

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

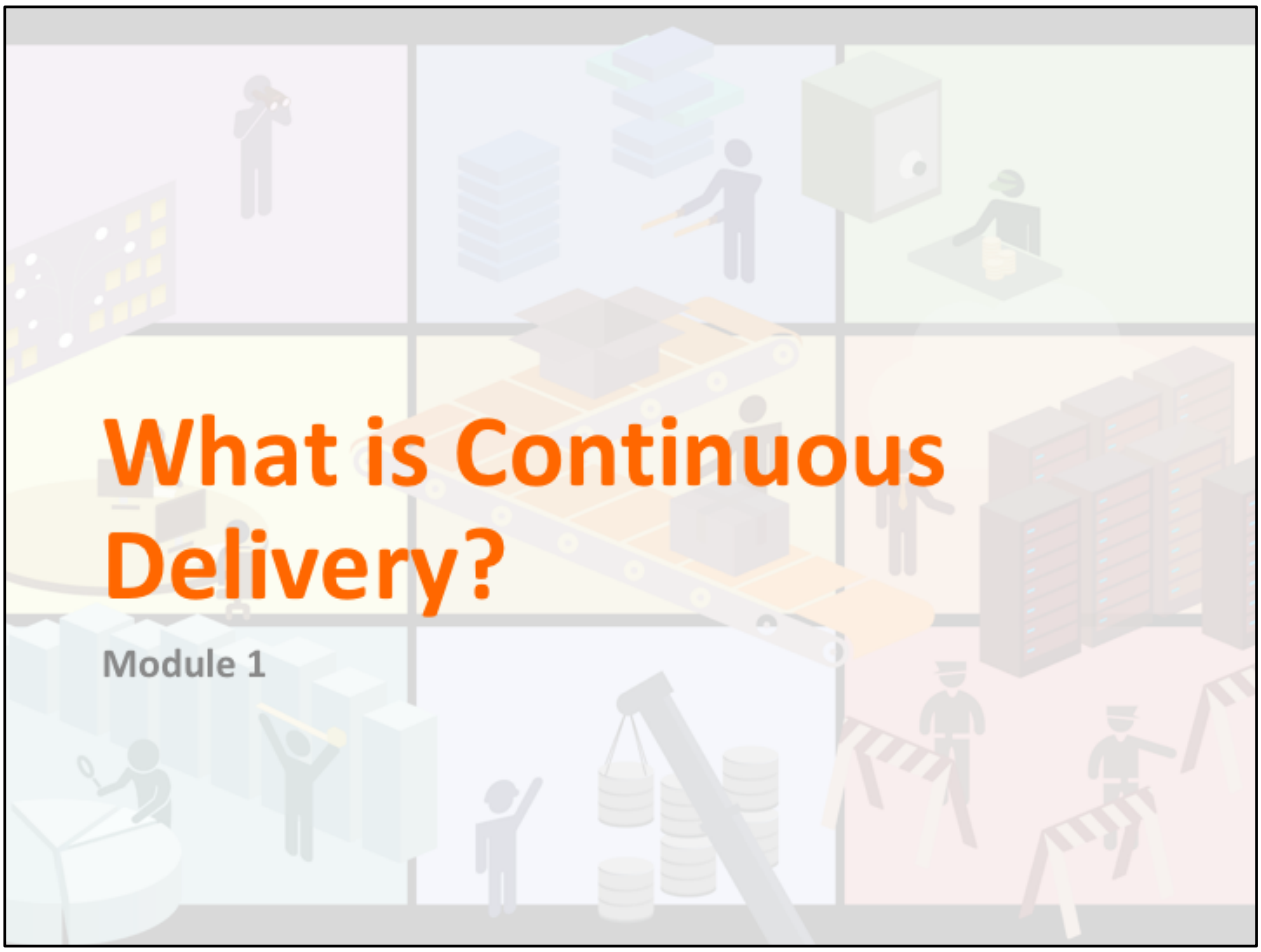


How do we compare to Agility-focused Certifications (e.g. PMI-ACP)?
The PMI-ACP is an exam-based certification designed to validate the knowledge and experience of general agile practitioners not for any particular agile discipline (A common misconception is that PMI-ACP is designed only for project managers). On the other hand, ICAgile offers a complete roadmap from novice to mastery within a number of agile disciplines. The PMI-ACP® does not offer a roadmap to mastery for agile in general, or for an particular agile discipline. This being said, ICAgile acknowledges the PMI-ACP® as a valuable milestone within one's agile learning journey. ICAgile's learning objectives within the Fundamentals of Agile Track align with the knowledge areas required to prepare for the PMI-ACP exam. In fact, a number of our training providers have accredited their PMI-ACP prep classes to offer the ICAgile Certified Professional Certification.

How are we compatible with Framework-focused Certifications (e.g. CSM, SA, PSM)?
ICAgile Certifications are based on learning objectives that were developed and reviewed by Agile experts from around the world. In designing the learning objectives the experts addressed one main question, "What do people need to learn to become agile experts?". This means that the learning objectives were designed to focus on applying an agile mindset to any discipline (i.e. Coaching, Development, Testing, Analysis, Management, Marketing, Leadership) regardless of the agile methodology. An ICAgile Accredited course may cover the [fundamentals of agile] learning objectives by teaching the agile mindset and Scrum while another accredited course may cover the same fundamentals of agile learning objectives by teaching the Agile mindset and Kanban. ICAgile has a rigorous process of reviewing course learning objectives to ensure the essence of agility is captured in every course.

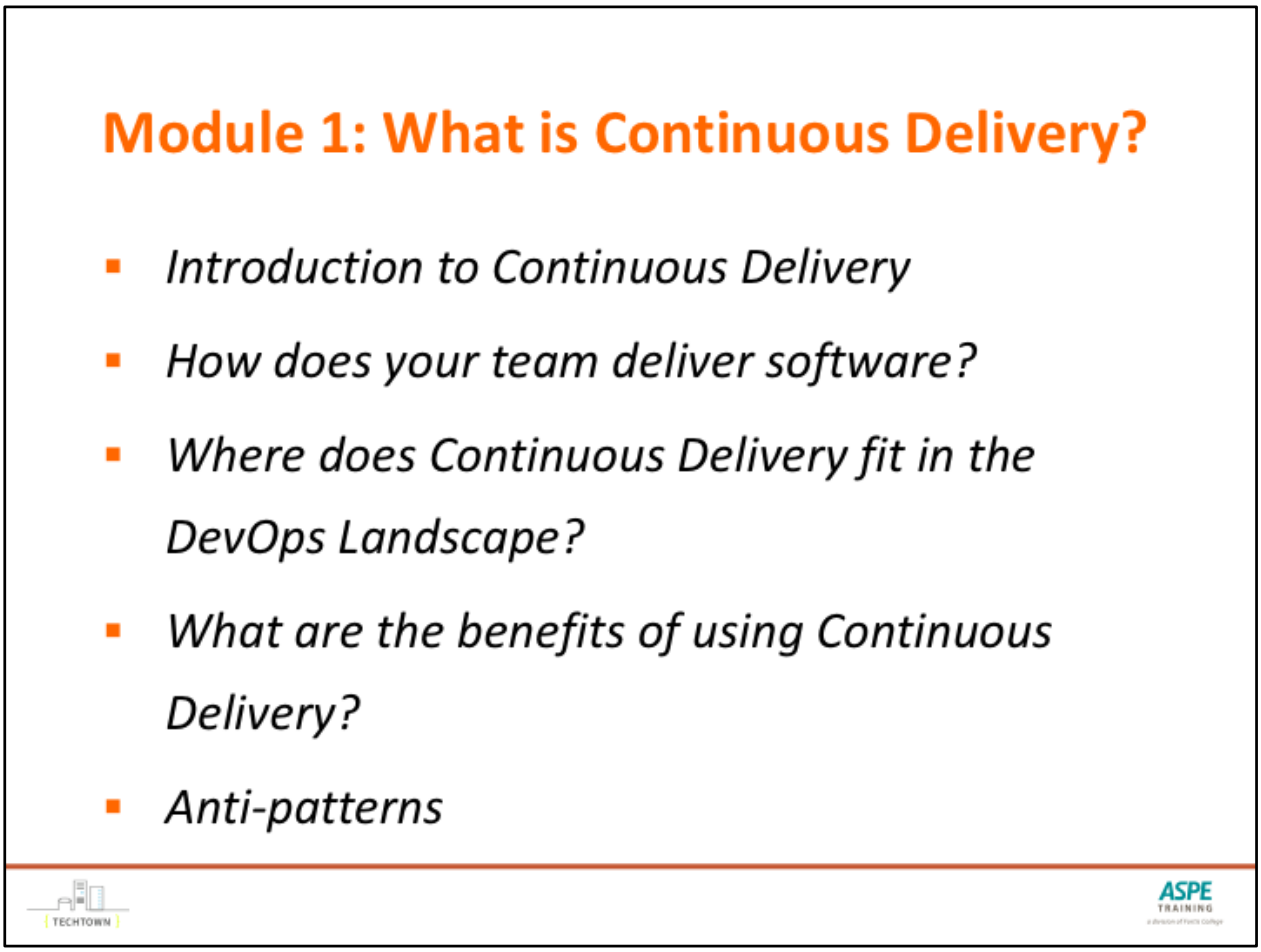
ICAgile Certifications are different from other certifications in the following areas:
1. ICAgile has 21 certifications covering 8 agile disciplines which are part of a learning roadmap.
2. ICAgile provides 3 levels of certifications: Professional-Level Certifications, Expert-Level Certifications, Master-Level Certification.
3. ICAgile's Professional level Certification is not earned by passing a standardized exam, instead, ICAgile requires training providers to have some form of meaningful assessment during the class (e.g. simulations, group exercise, etc.)
4. ICAgile's Expert and Master level Certifications are earned by performing a live demonstration to a panel of agile experts.
5. ICAgile has a rigorous accreditation process for all educational content ensuring its alignment to the learning objectives and encouraging active learning opportunities such as exercises, games, discussions and hands-on experience.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Continuous Delivery

"Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time."

– *Martin Fowler*

Ready to Deploy at Any Time?

Here are some of your main considerations:

- Minimize Branching and Merging
- Build Completes Successfully
- Units Tests Pass
- Acceptance Tests Pass
- Load and Stress Tests Pass
- Deployment Automation in place

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Introduction to Continuous Delivery

- Ready to deploy at any time
- Able to deploy any version
- Able to deploy to any supported platform

- *It doesn't mean you have to deploy continuously – it means you perform your engineering so you can if you want to*

Deployment should be fast, like 10 times a day. So we are always ready to deploy.

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Able to Deploy at any time

Deployment should be fast

- Less dependencies
- Less Hand-offs
- No long lived branches
- Maintain error free trunk
- Well tested
  - Automated unit test
  - Automated integration test

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Able to Deploy any Version

- Use low risk deployment architectures
  - Blue green deployment
  - Canary releases
- Store Deployment package in Version Control
- Store all required artifacts
  - Deployment Package
  - Deployment Script
  - Anything required to deploy and run the solution

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Deploy to any Platform

- Use of Configuration Management
- Platform management
  - Use of containers
  - Store supported OS .iso in Package Repository
- Store all required artifacts for supported Platforms
  - OS, Patches, Supporting Software and Utilities

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Continuous Delivery – "Classic" Core Principles

- "10+ Deploys per day"
- If it hurts do it more!
- Automation becomes a requirement
- Mean time to change decreases
- Mean time to recovery decreases
- Confidence increases and risk is reduced

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

You are succeeding with continuous delivery when:

1. Your software is deployable throughout its lifecycle
2. Your team prioritizes keeping the software deployable over working on new features
3. Anybody can get fast, automated feedback on the production readiness of their systems whenever somebody makes a change to them
4. You can perform push-button deployments of any version of the software to any environment on demand.

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Anti-Patterns

Some of the most common obstacles to effective CD practices are:

- **Manual Deployment**
- **Manual Testing**
- **Organizational Silos**
- **Manual Configuration Management**

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Value Stream Mapping

A **Lean method** for analyzing a Value Stream (or a system)

Value Stream: A series of steps that take a product or service from its beginning through to the customer.

Include
Waiting and **Process** time

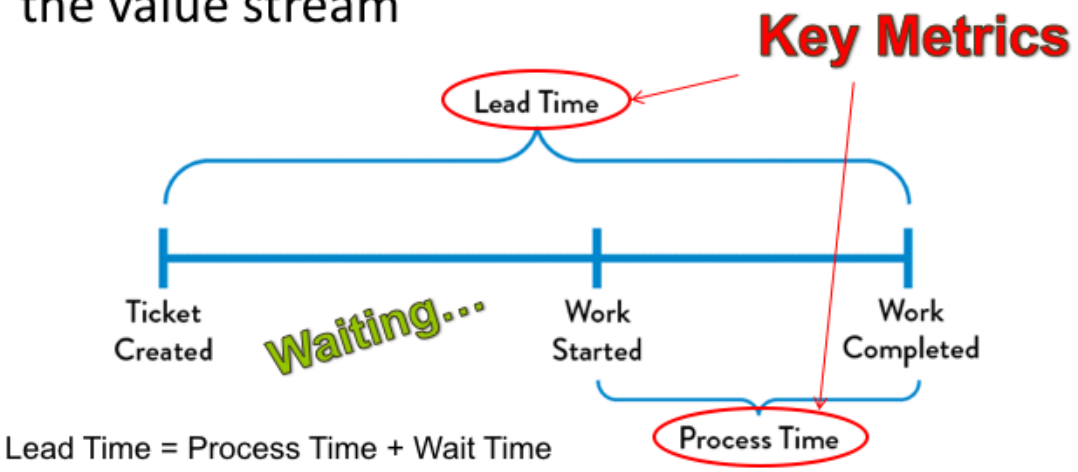
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Lead Time

The time required to complete a task:

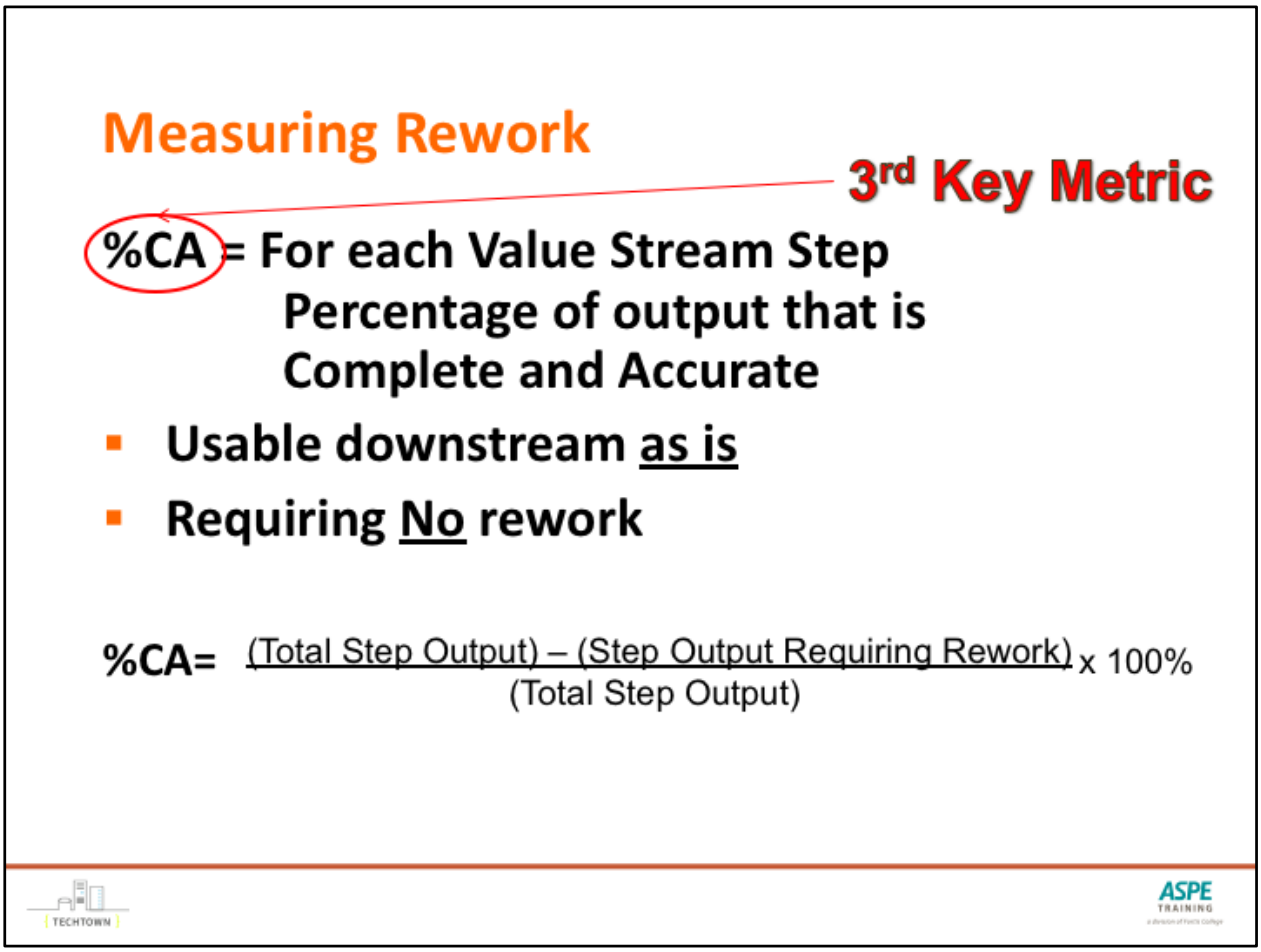
Task can be entire value stream or one step in the value stream



Lead Time = Process Time + Wait Time

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

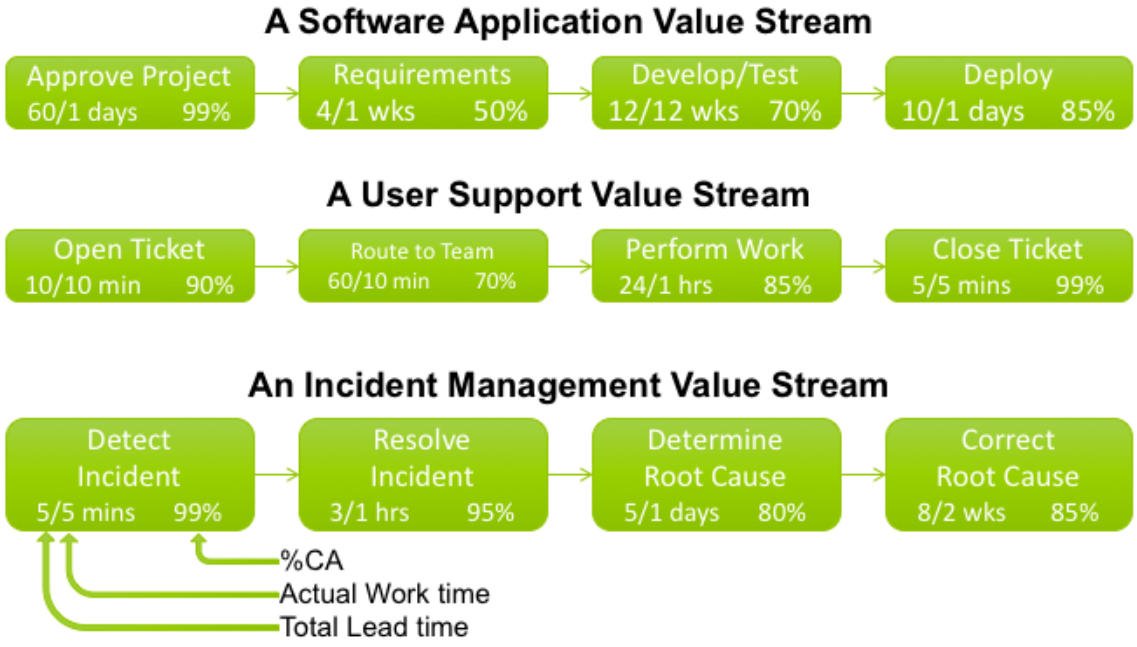


%CA = Percentage of work that is Complete and Accurate

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Value Stream Examples with Metrics



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise
Map Your Current Deployment Process

- **Create a Value Stream Map for your deployment process as it currently is**
  - **Draw a diagram of the Value Stream**
  - **For each step of the Value Stream, make your best estimate of the three key metrics**
    - Total Lead time
    - Actual Work time
    - %CA (Percent Complete and Accurate)

TECHTOWN

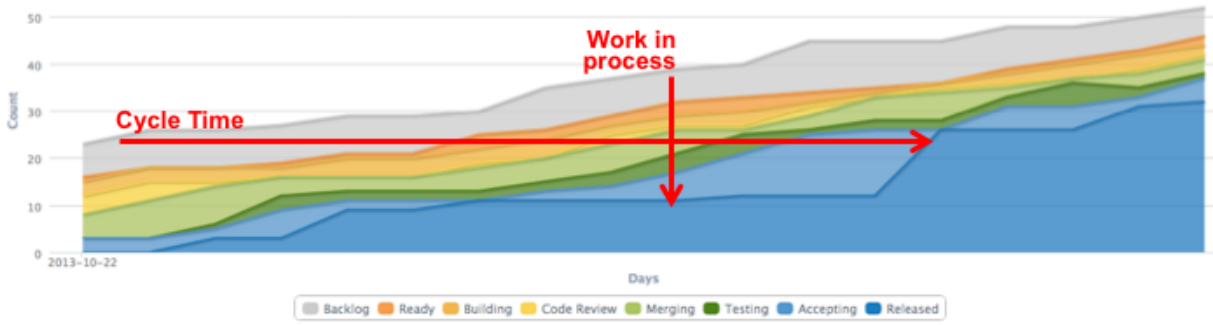
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Cumulative Flow Diagram

Like a Burn-Up chart, but not just totals



Bottleneck Activities:

- Excessive WIP
- Excessive Cycle Times (e.g. slow moving items)
- Uneven Growth

Where are the probable bottlenecks in this example?

TECHTOWN

ASPE TRAINING

NOTES:

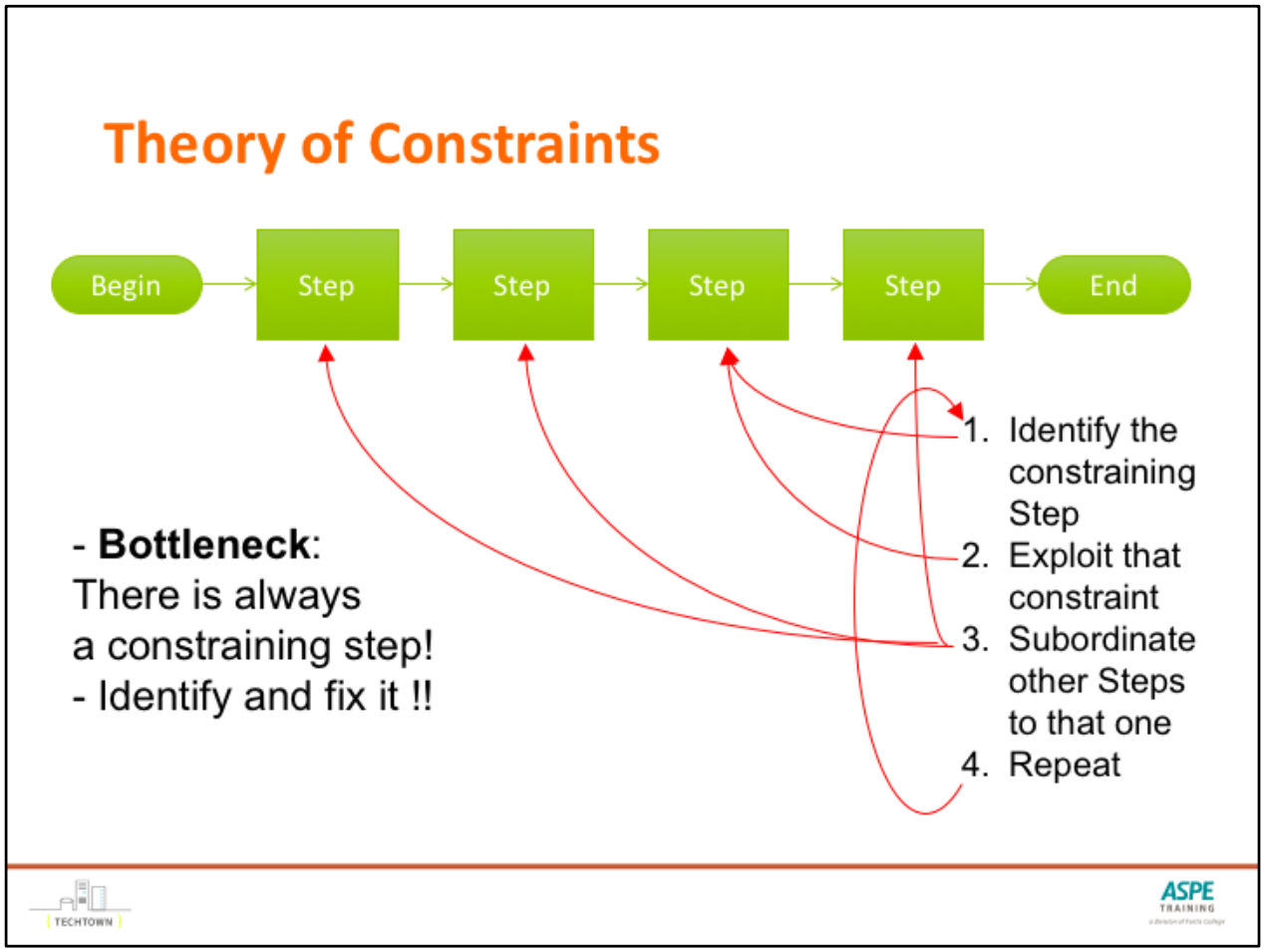
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Identifying the constraining steps

- Look at the cycle and WIP graphs
- Identify the constraining steps
  - Steps that takes lot of time
  - Steps that are repeated but can be automated
  - Steps that pass on errors downstream

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



1. The constraining step might constrain the process in various ways (adding wait time, adding processing time, adding defects, etc.)
2. Exploiting the constraint might mean providing more resources, automating it, or changing how it is done in some other way.
3. Subordinating other steps might involve de-optimizing them in order to make the constraining step more efficient, or providing more/different/better inputs for the constraining step.
4. After taking these actions, start over. Step 1 will tell you if the same step is still the constraint, or if there is now a different constraining step. (There will always be a constraining step!)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise: Identify Deployment Bottlenecks

- Based on your Value Stream Analysis:

  Identify the primary Constraints

  in your Deployment process

  - Manual
  - Long Running
  - Error Prone
  - Executed Frequently

TECHTOWN

ASPE TRAINING

Create

NOTES:

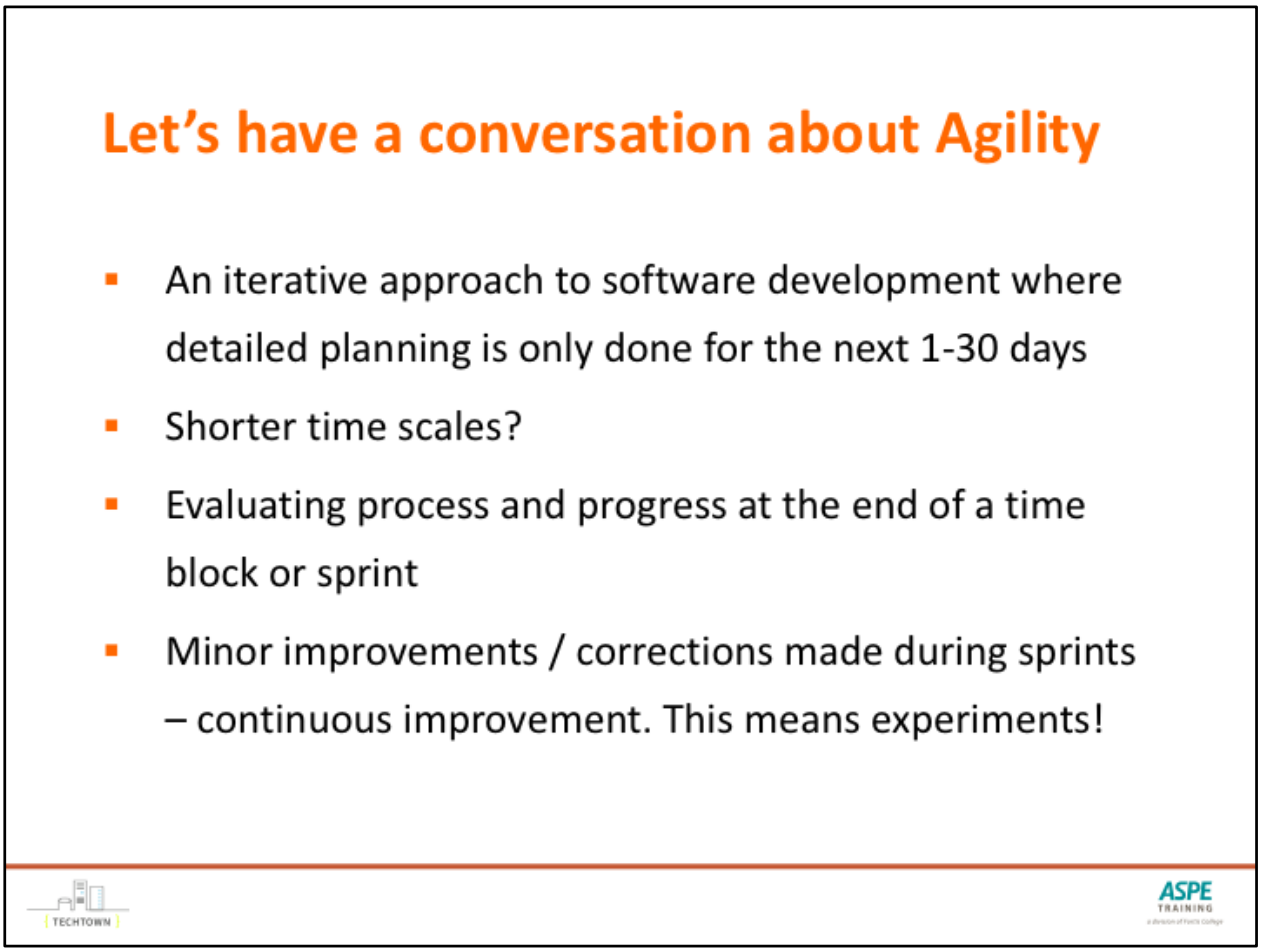
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

In order to implement Agile and DevOps principles, decision makers of any organization will need to understand the cultural shift that is necessary, and the complexities of successful adaptation. In this chapter, we will analyze the challenges of successful transformation, and how to deal with them.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



What is Agile?

Agile came about as a "solution" to the disadvantages of the waterfall methodology. Instead of a sequential design process, the Agile methodology follows an incremental approach.
Developers start off with a simplistic project design, and then begin to work on small modules. The work on these modules is done in weekly or monthly sprints, and at the end of each sprint, project priorities are evaluated and tests are run. These sprints allow for bugs to be discovered, and customer feedback to be incorporated into the design before the next sprint is run.
The process, with its lack of initial design and steps, is often criticized for its collaborative nature that focuses on principles rather than process.

Advantages of the Agile Methodology

1. The Agile methodology allows for changes to be made after the initial planning. Re-writes to the the program, as the client decides to make changes, are expected.
2. Because the Agile methodology allows you to make changes, it's easier to add features that will keep you up to date with the latest developments in your industry.
3. At the end of each sprint, project priorities are evaluated. This allows clients to add their feedback so that they ultimately get the product they desire.
4. The testing at the end of each sprint ensures that the bugs are caught and taken care of in the development cycle. They won't be found at the end.
5. Because the products are tested so thoroughly with Agile, the product could be launched at the end of any cycle. As a result, it's more likely to reach its launch date.

Engaging the Customer

- Agile Value: Customer Collaboration
- Agile Principle: Satisfy the Customer

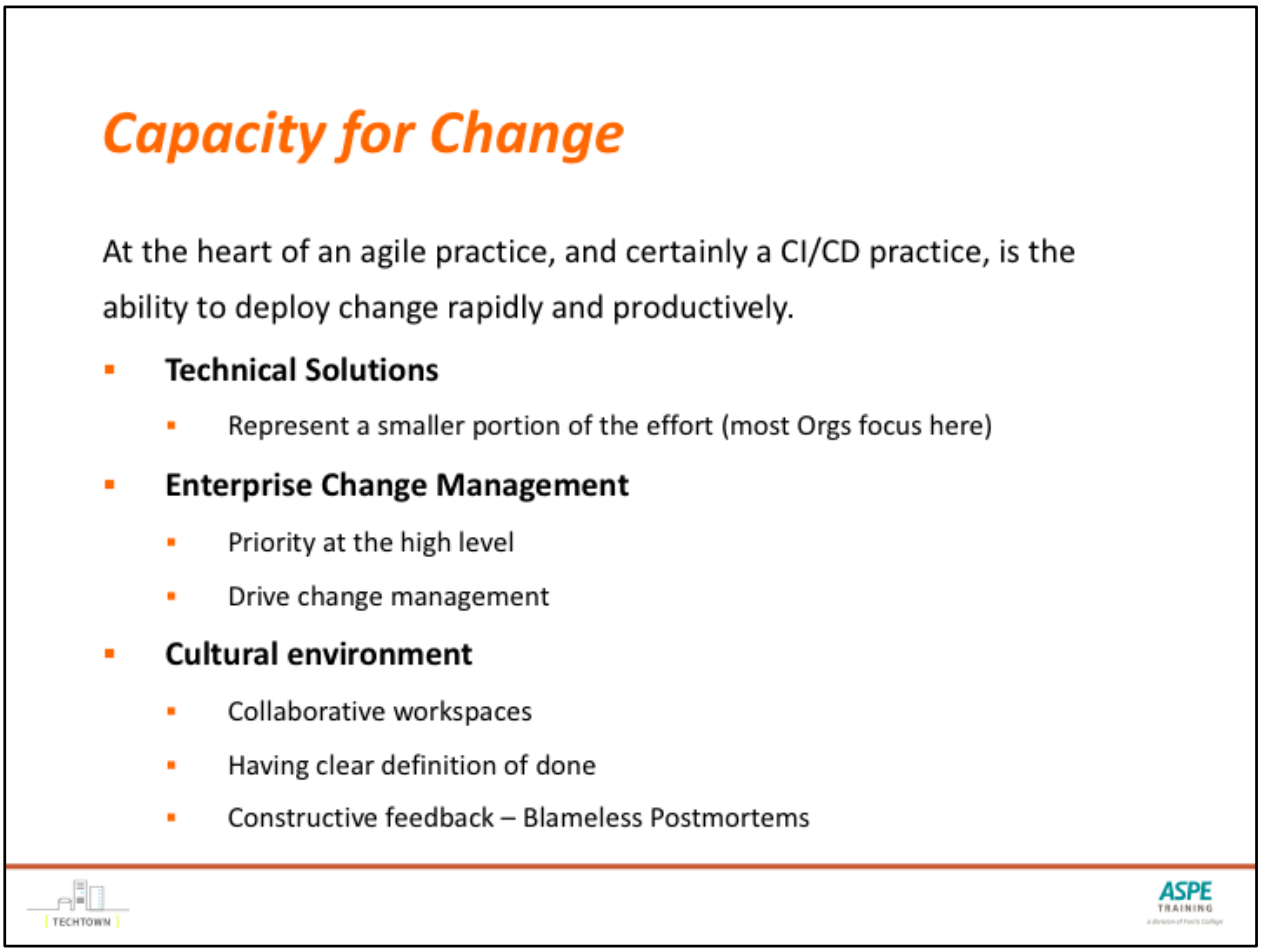
Engaging the customer:

- During development
- Before deployment
- After deployment

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Technical Solutions

- Required
- Represent a smaller portion of the effort
- Unfortunately, most organizations focus here

Enterprise Change Management

- Required Technical Process Changes
- Qualified Code Drops
- Small, Frequent integrations
- Prioritized Backlog

Shifting Culture:

- Team Objectives with design flexibility
- Collaborative workspaces
- Having a clear definition of “Done”

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Do you have a plan for always deployable code?

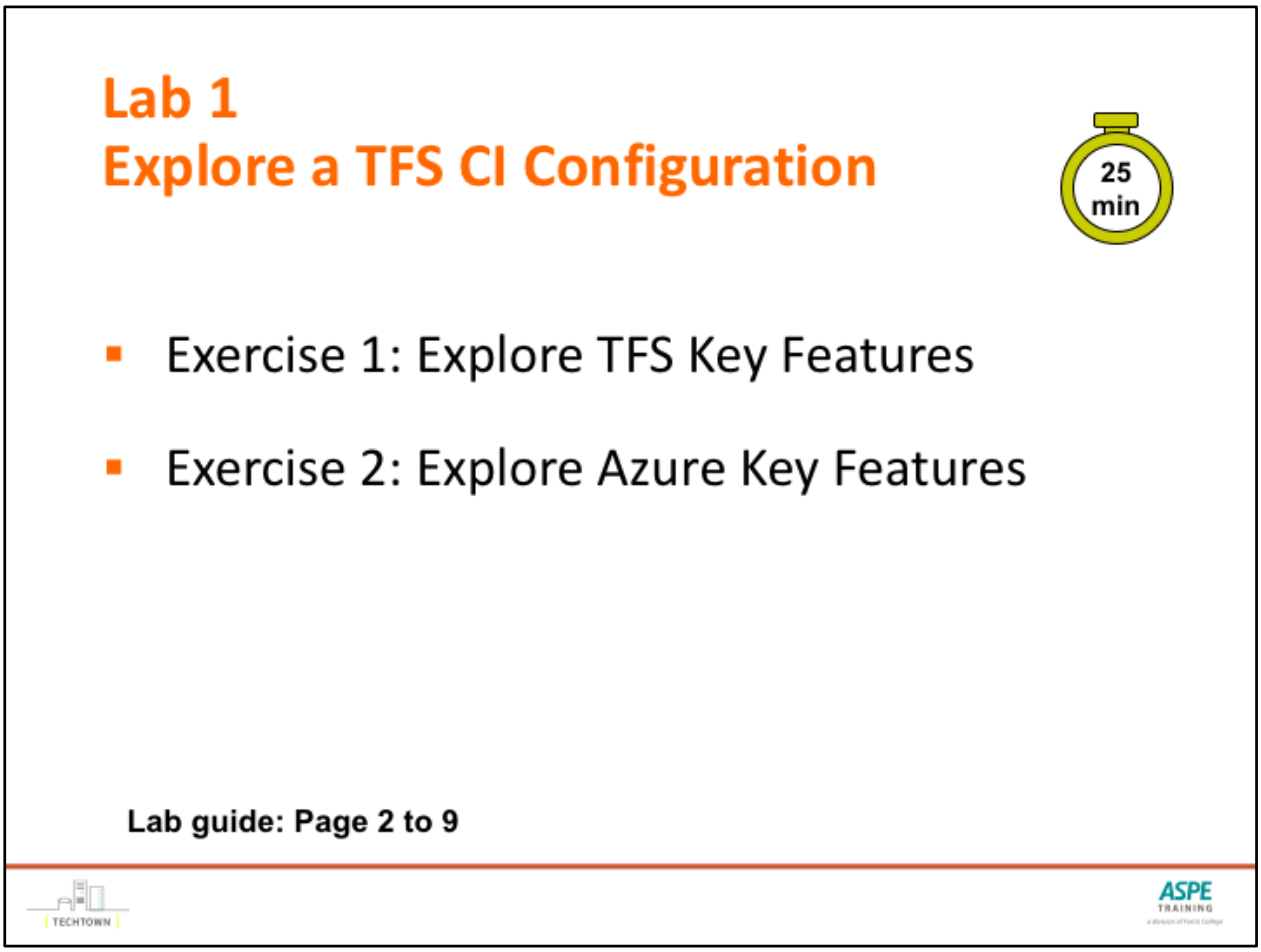
- Agile Principles at Scale
- Developers have a lot of input in defining team-level processes
- Is security involved?
- What is your monitoring/telemetry strategy?

Continuous Improvement requires everybody to have ownership of success

ASPE TRAINING

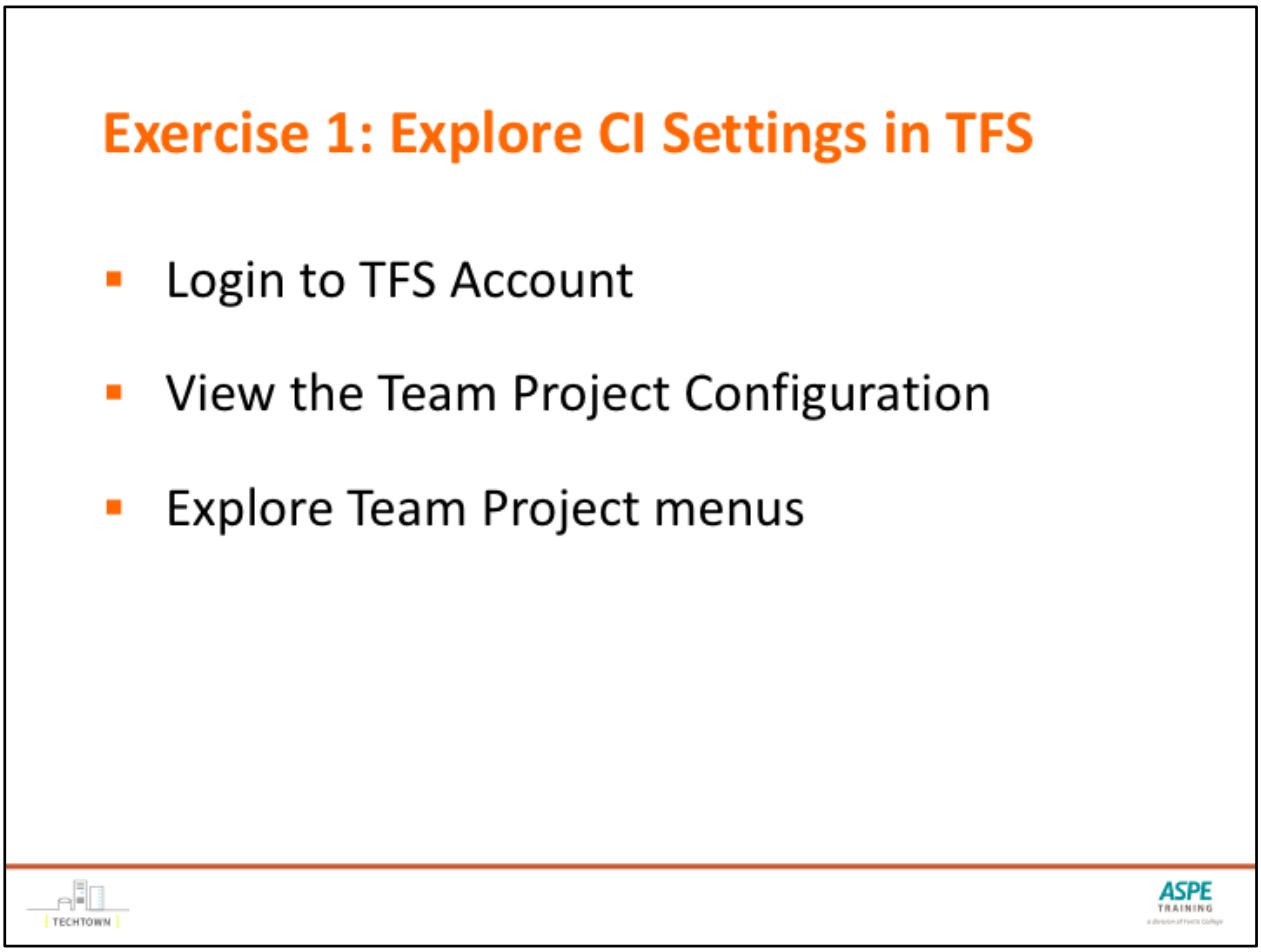
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In Lab 1 we will explore key features of Team Foundation Services including Work Item tracking, Version and Build / Release Automation.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 1: Explore CI settings in TFS

In this exercise, we will set up an account in **Team Foundation Server** (**TFS**),choose our template, and take quick tour of the various menu items that host different facet's of **TFS** functionality.

Login to TFS Account

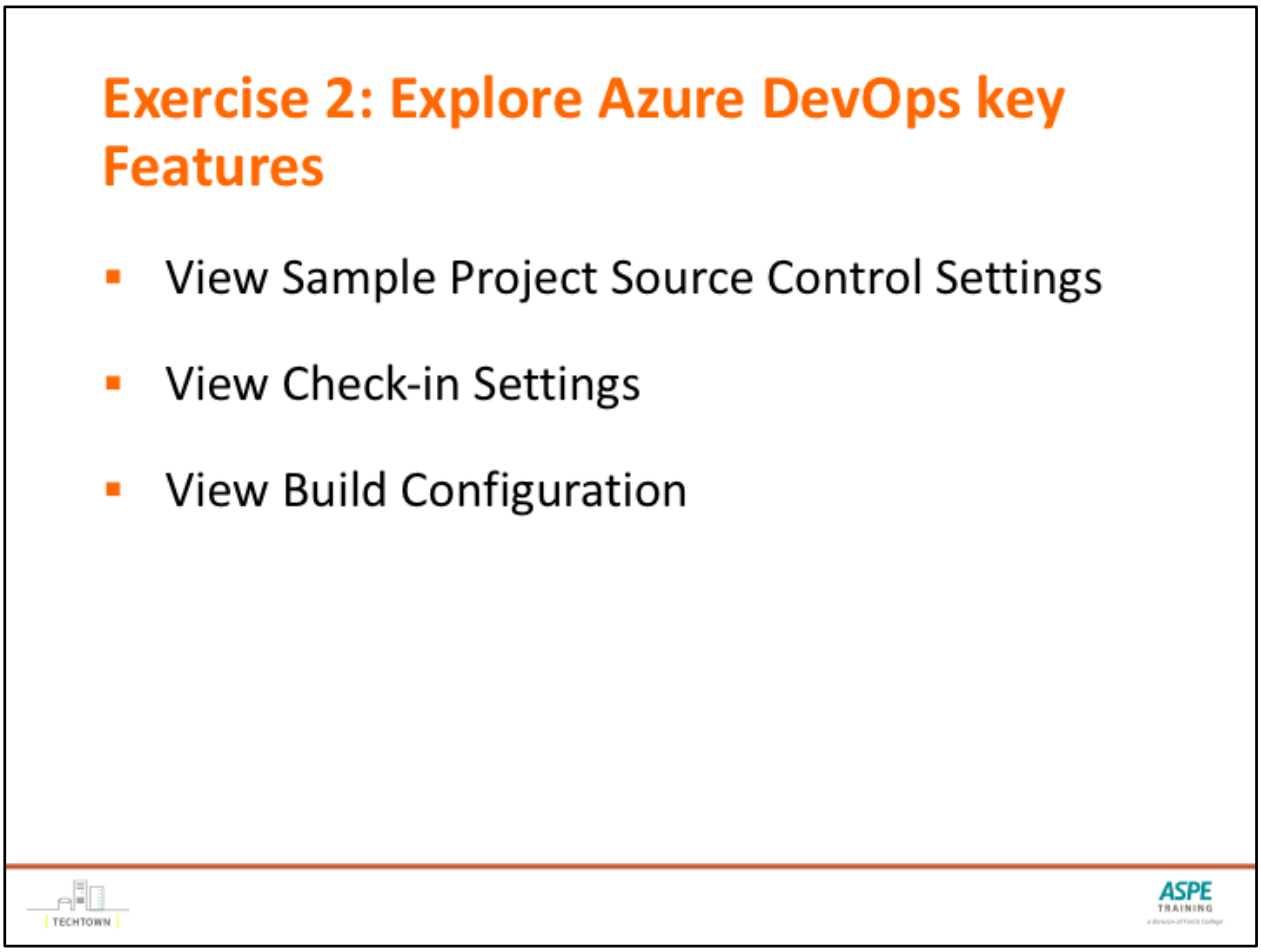
Navigate to **http://devopsclassroom.visualstudio.com**
Sign in as DevOpsStudent@Outlook.com
Password: JustM300

Create a New Team Project Configuration

Click the Azure DevOps Logo in the Upper Right hand corner of the webpage
Create a new project by clicking the **New Project** button*
Name your new project **Service Reservation XX** where XX is your initials, choose **Team Foundation Version control** for the Version control, choose the **Agile Work item** process, and click **Create**

*Note: If you do not see the New Project Button click the Visual Studio Icon in the upper left hand corner of the browser window

Continued on next page…



Exercise 2: Explore Azure Key Features

In this exercise, we take a brief look at our dashboard options, the code repository where we will be checking in our work, and the configuration areas which will allow us to define our builds and create tests.

View Sample Project Source Control Settings

1. Click on the **Dashboards** tab, and notice the tab in the lower right corner that has an option to open the project in **Visual Studio** in the lower right corner.

View Check-in Settings

1. Click on the **Code** tab. This is where the source code repository control where you upload code & check it into source control.
2. Click on the **Work** tab, and click the **Backlogs** menu. This is where we add new **User Stories** to our product backlog. This is also where we these **User stories** to the iterations of our sprints.

View Build Configuration

1. Click on the **Build and Release** tab. This is where we will create build definitions.
2. Click on the **Test** tab. Here, you can create **Test plans**, do **Test runs**, and **Load tests**.

Azure DevOps Documentation

https://docs.microsoft.com/en-us/azure/devops/pipelines/?toc=%2Fazure%2Fdevops%2Fpipelines%2Ftoc.json&bc=%2Fazure%2Fdevops%2Fboards%2Fpipelines%2Fbreadcrumb%2Ftoc.json&view=azure-devops

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Defining Continuous Delivery

Continuous Integration – Continuous Delivery can not be applied without first implementing Continuous Integration. Processes and protocol will be different for each organization, and establishing a base level of stability will be essential. ***Points for discussion:***

- Source Code Management
- Automatically triggering a build
- Integrating often
- Identify Code conflicts as early as possible

Scripted Environments – Scripted environments work differently from most traditional organizations, where people log onto server and make changes to get applications to work properly. When environments are scripted, it makes the triage process less difficult, and also ensures that servers have the same configuration across the system. ***Points for discussion:***

- Creating a Script
- Virtual Machines
- Configuring settings from OS to Container

Scripted Deployment – Once servers have been configured across an entire system, a process for automating the deployment across these environments should be applied. This process is commonly refereed to as scripted deployment. ***Points for discussion:***

- Automating Deployment
- Deploying Applications
- Configuring Applications
- Server to server repeatability

The Necessity of Continuous Integration

Evolutionary Database Design – EDD is an important approach to keeping your application in an "always releasable" state. Basically, it focuses on managing database changes while ensuring that schema changes don't break the application. ***Points for discussion:***

- Managing Database Changes
- Ensuring that Changes don't break the application
- Automatic Database Updates

A Deployment Pipeline – The deployment pipeline defines how new code is integrated into the system, promoted through though stages of environment, and serves as a progression model for software changes. ***Points for discussion:***

- Integrating new code into the system
- Static Code Analysis
- Unit testing at the component level
- Regression testing
- Non functional testing (Performance / Security)

Orchestration – The orchestrator tool coordinates automation, enables scripted environments, and moves validated versions of applications and scripts forward to the next stage of the development pipeline. ***Points for discussion:***

- Coordinating Automation
- Creating new environments on virtual machines
- Scripted deployment
- Database Schema Updates

Continued on next page…

Continuous Delivery Necessities

Continuous Delivery Architecture

CD architecture defines the different pieces of the continuous delivery pipeline that eventually come together. The object oriented approach works well for this, as it prevents scripts from becoming big, long, monolithic items that are fragile and hard to maintain. ***Points for discussion:***

- Core pieces of Continuous Delivery
- Scaling Processes across large organizations
- Using an Object Oriented approach
- Avoiding complexity in basic scripts

Scripted Environment Architecture

Scripted environments will be a necessary element when it comes to successful transformation to Agile development. Executives will need to ensure these technical changes happen because they start the cultural change that is critical in bringing Development and Operations teams working with common objectives. ***Points for discussion:***

- Leveraging common scripts across deployment stages
- Defining the Difference between Environments
- Developers & Operations using the same tools for common objectives

Post Deployment Validation

Post Deployment validation is a key principal of the scripted deployment process. Validating individual success for every server will help do define and fix all issues before deploying code. ***Points for discussion:***

- Deploying Code Across Multiple Servers
- Post Deployment Validation Tests
- Differentiating between Code & deployment issues
- Validating individual Servers

NOTES:

Continuous Delivery Implementation:

Embracing the Cultural change

Applying DevOps principles at scale of enterprise solutions is going to require implanting Continuous Delivery. Embracing Continuous Delivery implementation across an organization is a must, because it will help to align business objectives between Developers, Operations, and executive level planning. ***Points for discussion:***

- Coordinate work across teams
- Provide Feedback to Developers
- Let Developers see their features work in production

Using Business Objectives to prioritize implementation

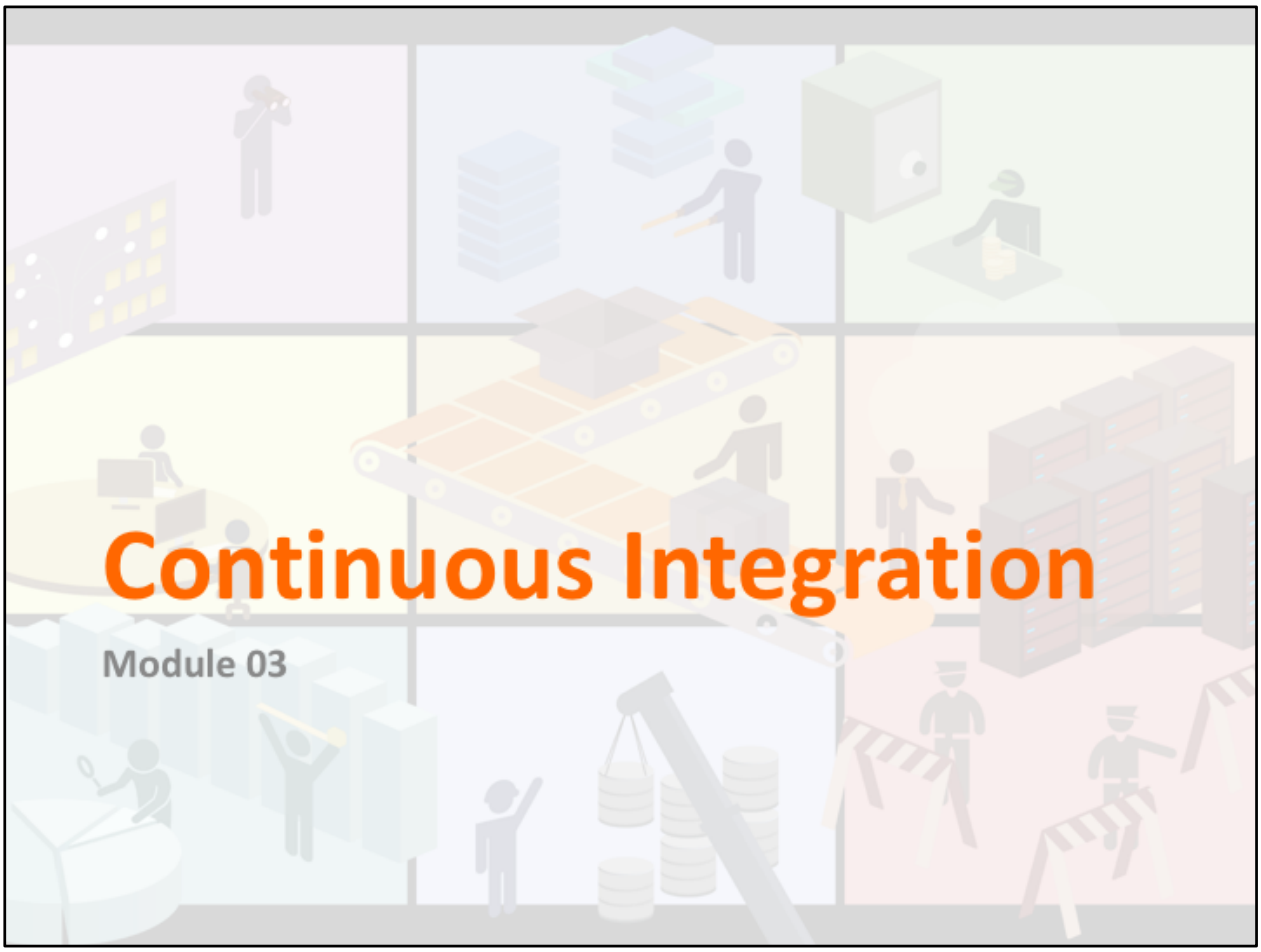
Common goals force everybody to focus on immediate practices that affect customer value and speed to market, and once this is established it will continue to improve. ***Points for discussion:***

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

- Increase frequency of deployment
- Determine pain points
- Use the pain of increasing frequency prioritize changes

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

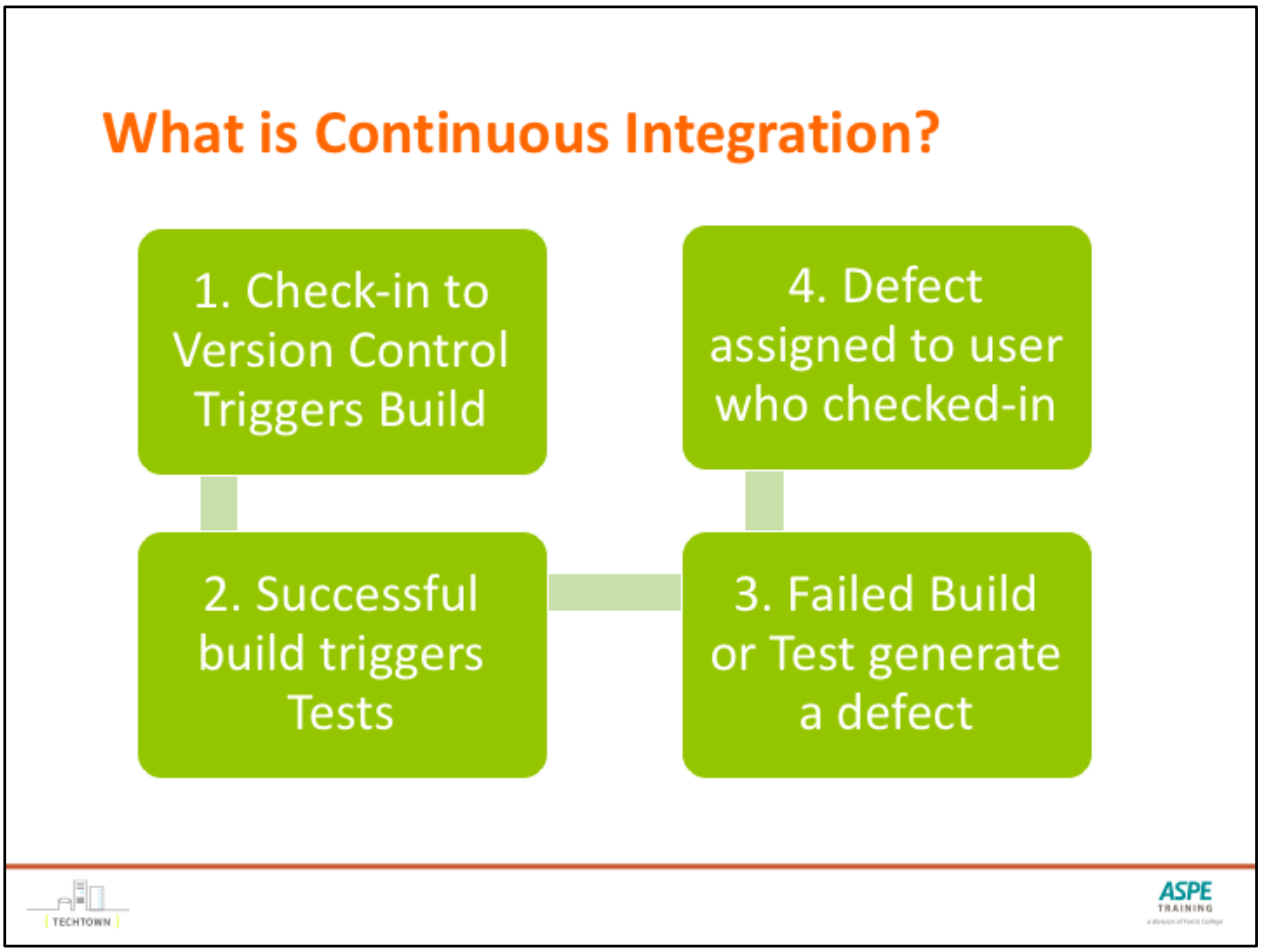
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Module 3: Continuous Integration

- What is Continuous Integration?
- Where does Continuous Integration fit in the DevOps Landscape
- How does Continuous Integration work?
- Continuous Integration Essentials
- Common Continuous Integration Practices
- Benefits of Continuous Integration
- Continuous Integration to Continuous Delivery

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

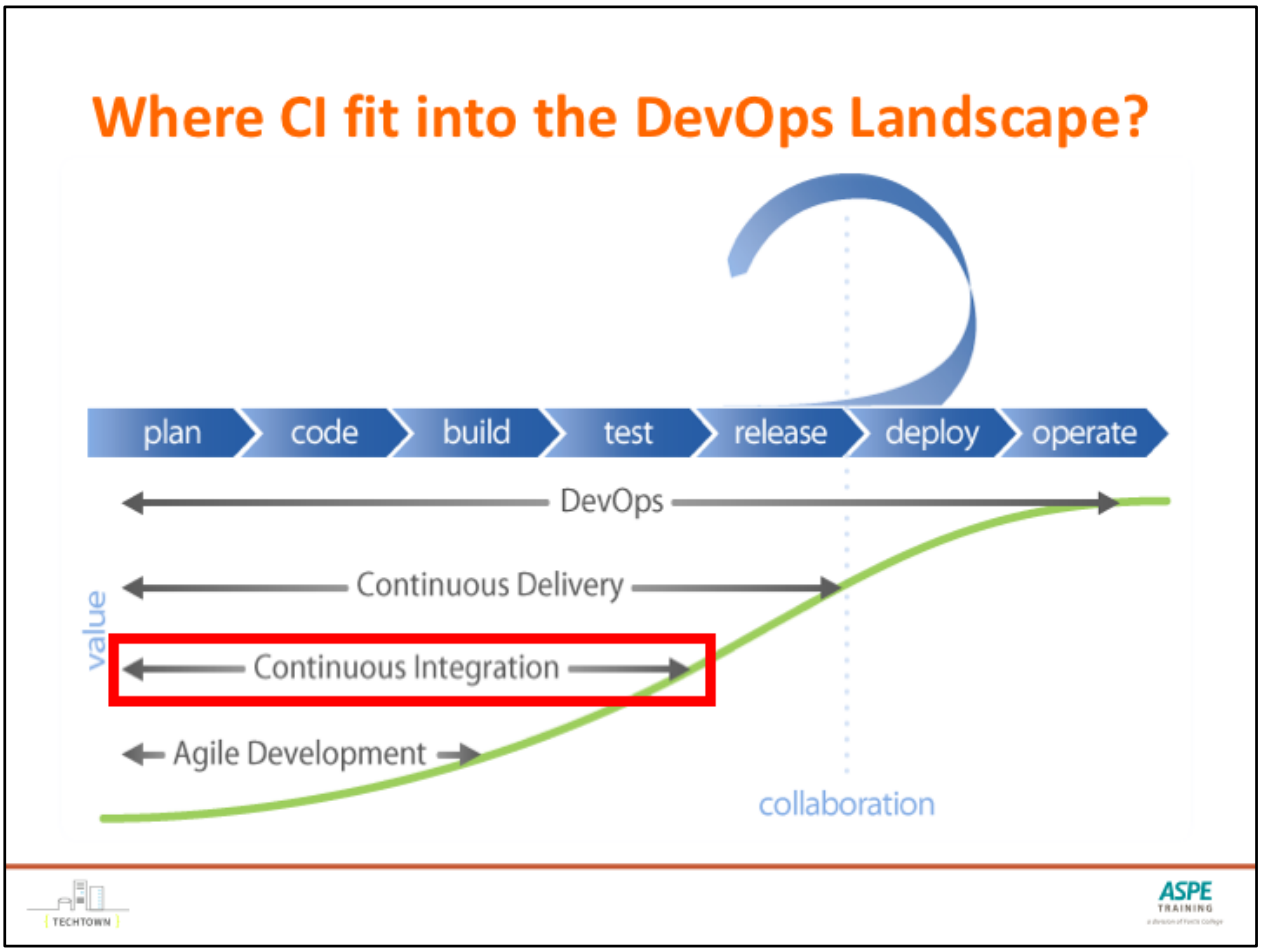


Continuous Integration (**CI**) is a software development practice in which isolated changes are automatically compiled (build), tested and reported on when they are added to a larger code base. **CI** is an important component of **DevOps** and **Agile** strategies which saves time, improves software quality, and increases speed to market for product delivery. These strategies cannot be incorporated successfully without a seamless execution of **Continuous Integration**.

Developers who practice **CI** develop their team solutions one piece at a time. When they are finished with each small part, it is submitted and tested so that bad code does not get built into existing functional software. As **Agile** developers integrate incremental updates, and fixes, each change is checked in and documented in a centralized **Source Code Repository** like **Team foundation Server**. When a piece of code is submitted, the functionality of each change is then verified through automated testing and virtual integration. Code clash, dependency issues, and other errors will be detected early, and eliminated before they become unmanageable.

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Continuous Integration Essentials

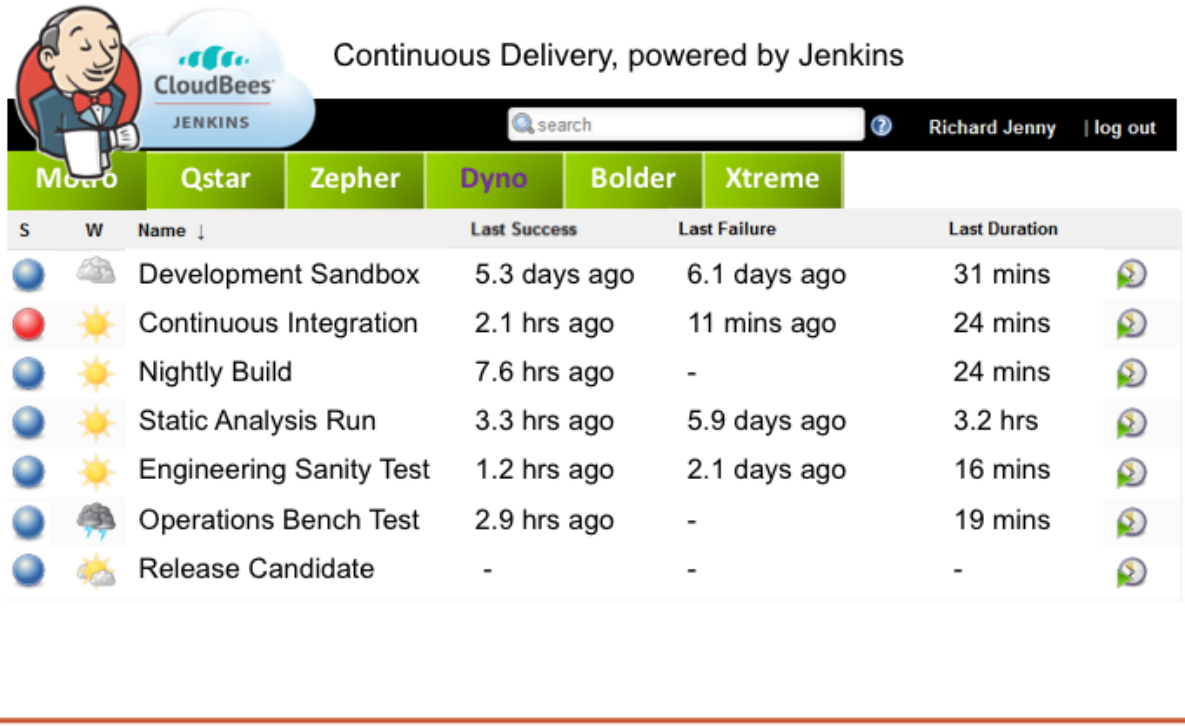
- Automated Build
- Automated Tests
- Automated platform creation
- Deployment Package Creation
- Deployment Package Storage

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Common Continuous Integration Practices



TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Benefits of Continuous Integration

- Feedback early and often
  - Integrated Build Results
  - Test Results
  - Focus on smaller task items and change set sizes
- If you are going to fail, fail early
- Enables automated testing
- A necessary milestone on the path to Continuous Delivery and DevOps

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Continuous Integration Vs Delivery Vs Deployment

- **Integration**: Integrate code into a main branch early and often, instead of building out features in isolation and integrating them at the end of a development cycle
- **Delivery**: Automate the software delivery process so that teams can easily and confidently deploy their code to production at any time; by ensuring that the codebase is always in a deployable state
- **Deployment:** Automatically deploys each build that passes the full test cycle; instead of waiting for a human gatekeeper to decide what and when to deploy to production

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

For the next lab (Code Coverage)

- Metric that can help to understand how much of the source is tested. Code coverage tools will use one or more criteria to determine how the code was exercised or not during the execution of the test suite.
  - **Function coverage:** how many of the functions defined have been called
  - **Statement coverage:** how many of the statements have been executed
  - **Branches coverage**: how many of the branches of the control structures (if statements for instance) have been executed
  - **Condition coverage:** how many of the boolean sub-expressions have been tested for a true and a false value
  - **Line coverage:** how many of lines of source code have been tested

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

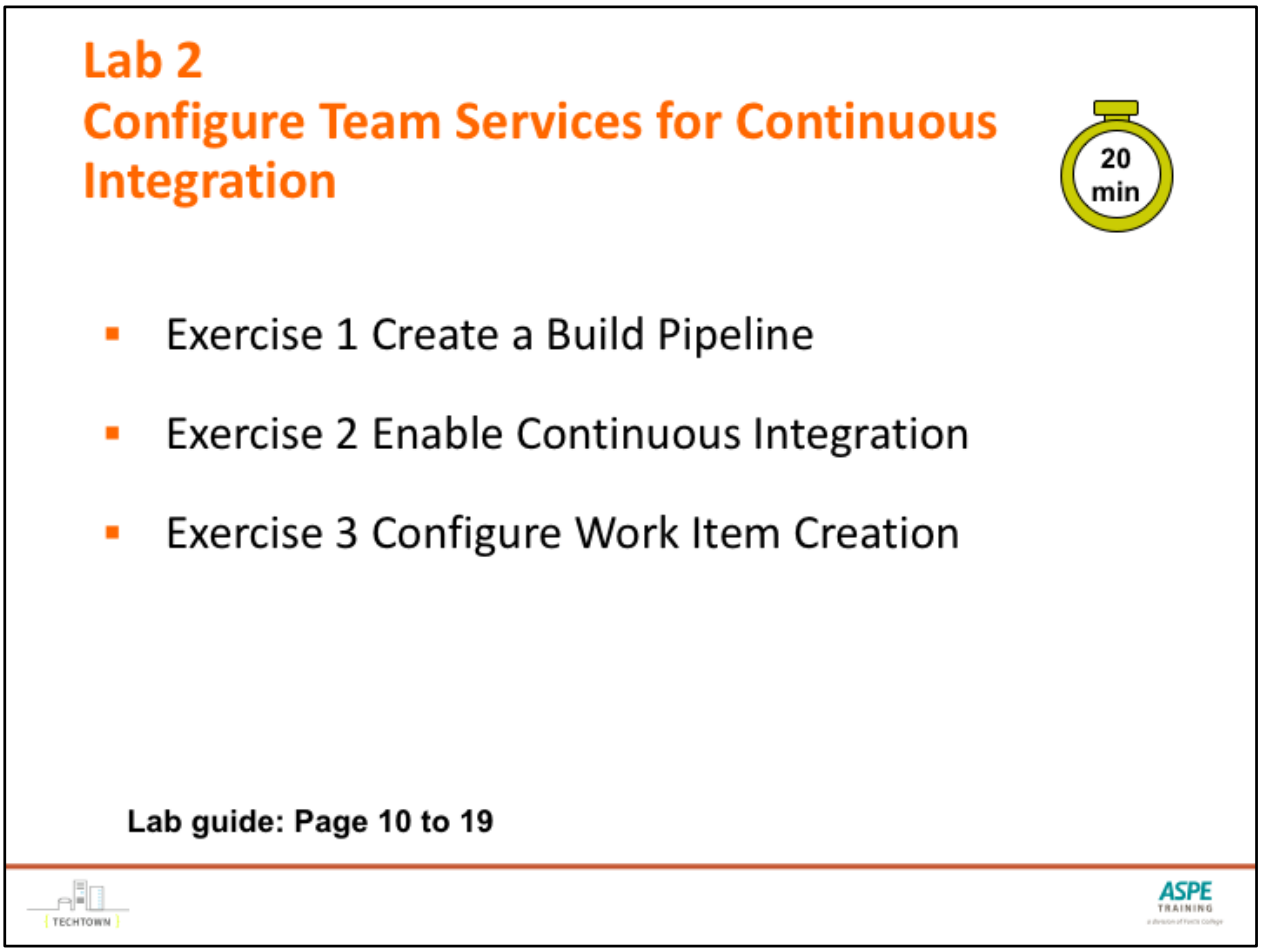
For the next lab (Triggers)

- **CI Trigger:** If a CI build fails due to bad code being committed, the code is still committed and reside in the source control
- **Gated check-in Trigger**: TFS creates a *shelveset* containing the code that's being validated, then runs a build of that code. Only if that code builds successfully and all configured unit tests pass does the code actually get committed
  - Benefits:- It prevents checking in bad code
  - Drawbacks:- Makes the process slower :- Others have to wait, until the code passes gated check-in trigger

ASPE TRAINING

NOTES:

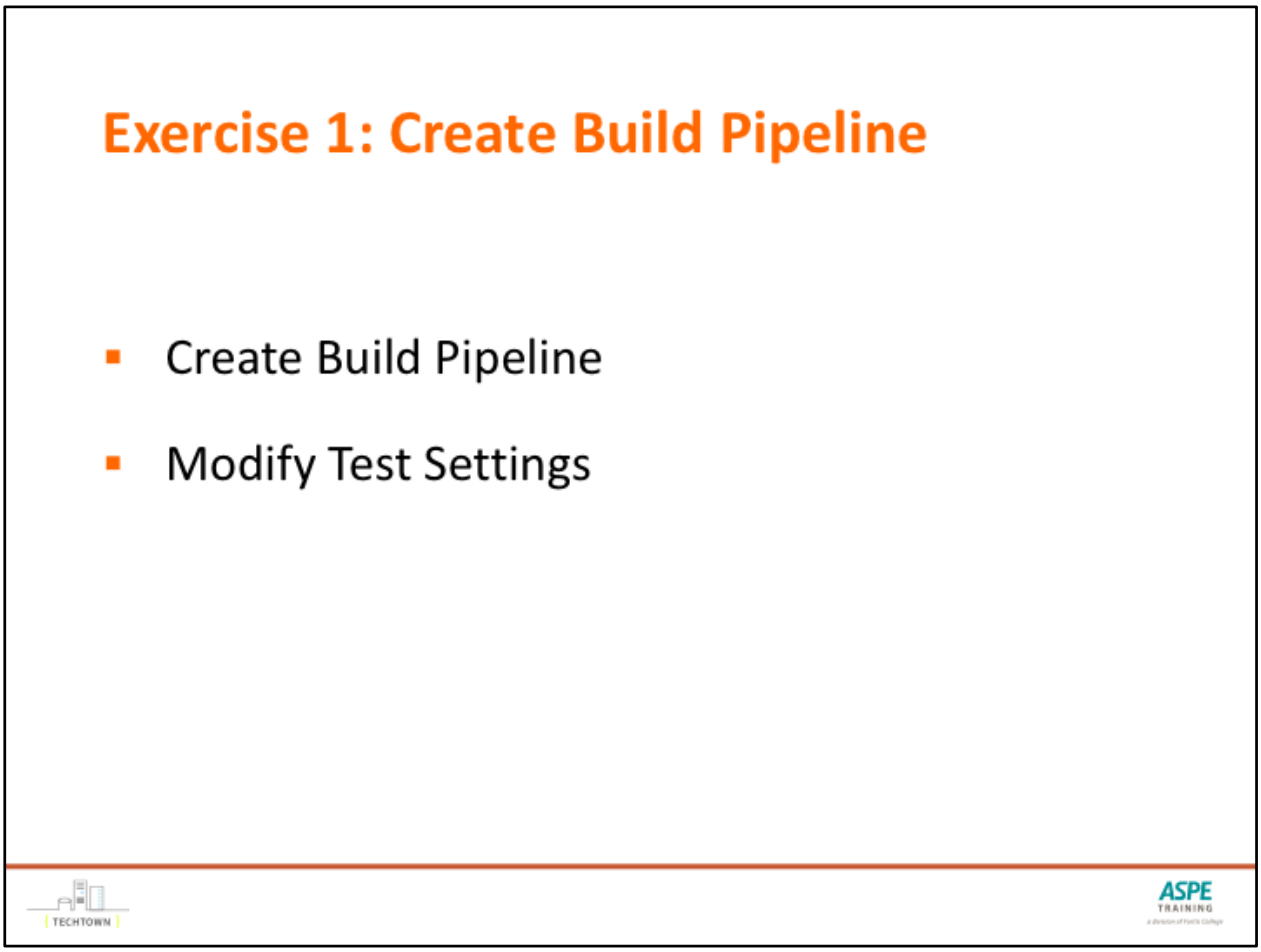
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Lab 2 – Configure TFS for Continuous Integration

In this Lab we will configure **Team Foundation Services** for Continuous Integration. We will also configure **Gated Check-Ins** to restrict code that does not build or does not pass all tests from being allowed into source control. In addition we will have the system automatically assign a new work item (**Bug**) to the user that checked bad code into Source / Version control.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 1: Configure Build Trigger

In this exercise, we will configure a Continuous Integration Build Trigger in **Team Foundation Services**.

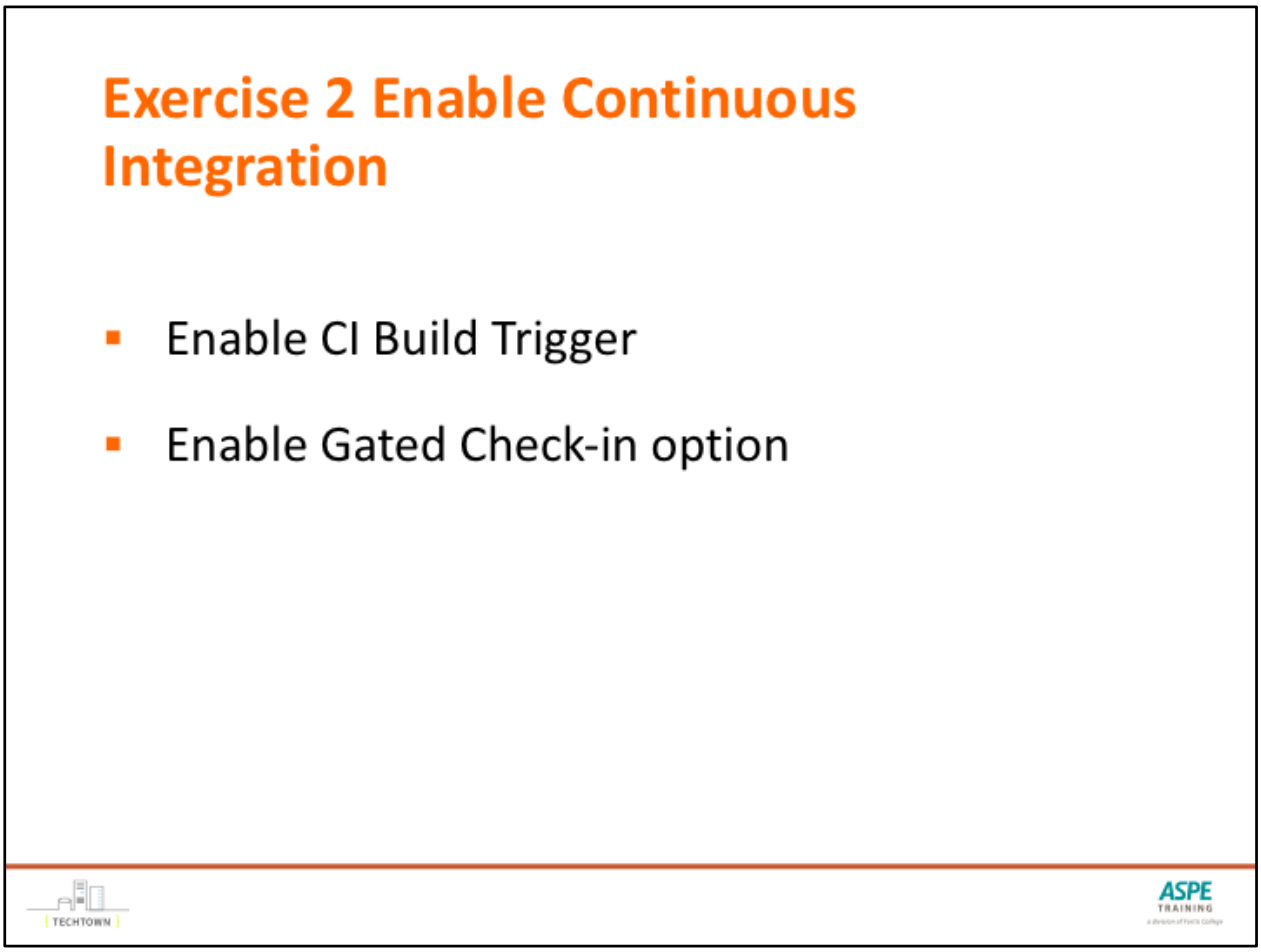
Create Build Definition

1. Open a browser & navigate to: https://DevOpsClassroom.VisualStudio.com/_projects
2. Login as User: DevOpsStudent@Outlook.com Password: JustM300
3. Select the **Service Reservations XX (XX=Your Initials)** project.
4. Inside of **Service Reservations** project, click on the **Pipelines** tab.
5. Click the **+ New** button in the top right corner.
6. Choose a new **ASP.NET** template, and click apply
7. The *Build Definition* will be called: **Service Reservations-ASP.NET-CI**
   *(Notice that we have an option to use the "**Hosted VS2017**" build)*

Modify Test Settings

1. Click on **Test Assemblies.**
2. Scroll down until you see the "**Code coverage enabled**" option.
3. Check the corresponding checkbox.
4. Click the dropdown next to **Save and Queue** and Select **Save** to save the **Build Definition.**

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In this Exercise, we will enable a build trigger for Continuous Integration, and ensure that bad code can not be checked in by enabling **Gated Check-in.**

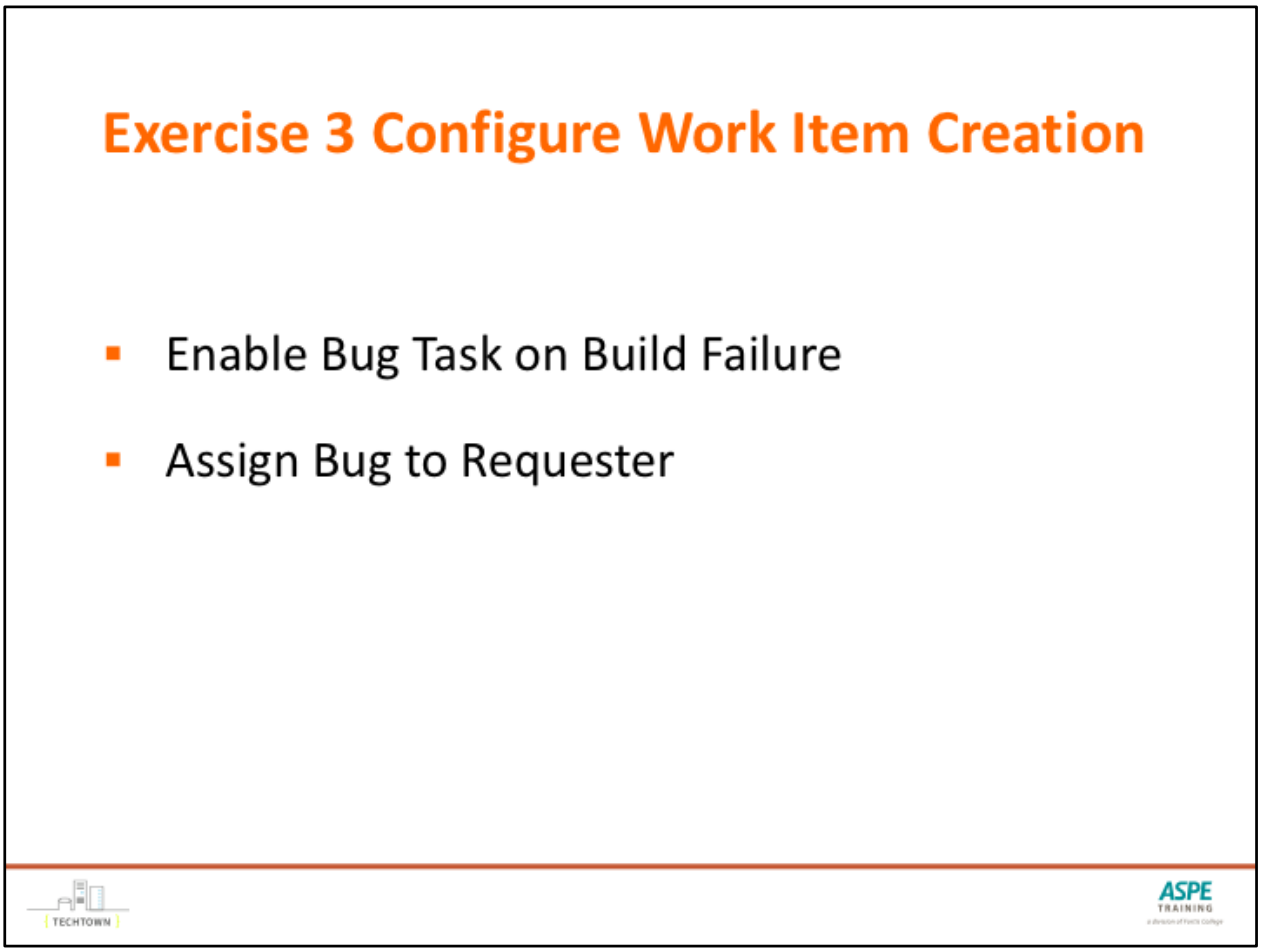
Enable CI Build Trigger

1. Navigate to the **Triggers** tab
2. Enable the Continuous Integration Trigger by flipping the switch under **Enable this Trigger** that says **"Disabled"** to the **"Enabled"** position.

Enable Gated Check-in option

1. Enable Gated Check-ins by flipping the switch under **Enable Gated Check-ins** that says **"Disabled"** to the **"Enabled"** position.
   (Notice the difference between the way it looks when it's switched on, as opposed to when it's switched off.)
2. Accept the rest of the defaults.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In this Exercise, we will set up a bug task which will be introduced upon build failure, and ensure that the bug is sent to the individual who checked in the code that caused the build to fail.

Enable Bug Task on Build Failure

1. Click on the **Options** tab.
2. Under options, select **Create work item on failure** by flipping the corresponding switch to "**Enabled**".
   *(Notice that the type of work item that will be created upon failure is **Bug**.)*

Assign Bug to Requester

Notice the **Assign to requester** checkbox is checked by default, so whoever committed the changes will be notified if any bugs appear.

1. Click the dropdown next to Save and Que and Select Save to save the **Build Definition.**

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise
Identify Deployment Improvements

- Update your Value Stream Map based to reflect using Continuous Integration
  - Estimate new Lead Times and %CAs
- Identify how CI improves your Deployment process

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Continuous Delivery and Your Environments

- Infrastructure Management
- Communicating with the Operations Team
- Configuration Management
- Infrastructure in the Cloud
- Infrastructure Maintenance

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Infrastructure Management

Infrastructure Challenges	Solution: Managing Infrastructure as Code
Cycle Time – Can take too long and therefore costly	Manage infrastructure as code for reliable, repeatable configurations
Manually versioning environments is costly	Ensure speedy and reliable deployments with a proven DevOps platform
	Manage all the complexity of the cloud by treating infrastructure as code.
	Enables continuous delivery practices like version control & continuous integration.

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Communicating with Operations Team

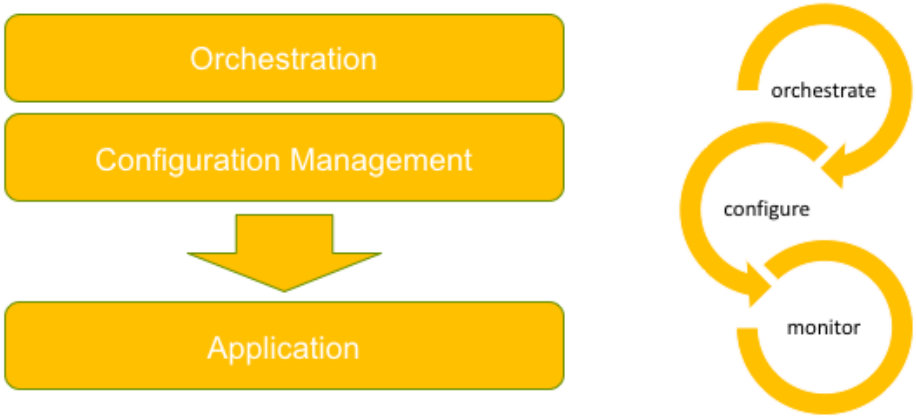
- **Lack of communication can lead to resistance to change**

Common objections to change	Common strategies to overcome resistance
• Because they fear the consequences, which are unknown • Because they do not trust • Because they believe it is a fad and not permanent • Because it threatens them in some way, perhaps by losing status or lack of skills • Because they do not know why the change is needed	• Establish throughout the organization that change is needed • Communicate clearly the reason behind the change • Offer support resources such as training when applicable • Be consistent, well researched, and follow through with any committed changes • Start with small victories and build on them

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Modern Automation Stack

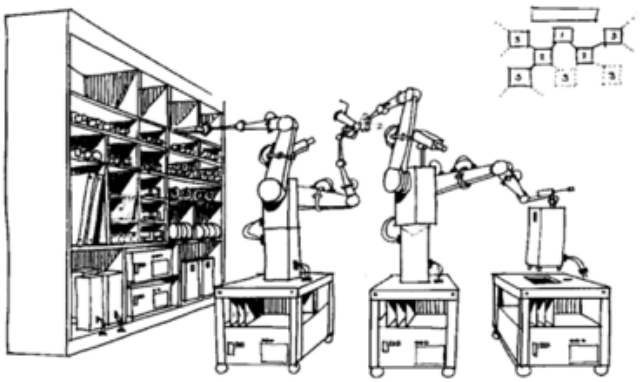


NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Orchestration

- Not the same as configuration management
- Infrastructure deployment
- Critical for infrastructure automation



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

We Automate, What/Why/Orchestrate…?

We have been automating things forever. Why is orchestration so hot these days?

What is different today is:

- Cloud computing, containers, Microservices, and horizontally scaling applications.
- The underlying tooling has changed, AND architectures are made up of many more moving parts than ever before.
- We need to dynamically build and tear down infrastructure on demand.

https://www.virtualizationpractice.com/the-race-to-own-orchestration-37184/

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Deployment vs Orchestration vs Config Management

'Deployment' is the process of putting a new application, or new version of an application, onto a prepared application server.

'Provisioning' is normally used by Ops to refer to getting computers or virtual hosts ready to use, and installing needed libraries or services on them.

- **The thing to remember is that 'deployment' does not, as a rule, include 'provisioning'.**
- Developers often use Chef, Ansible or Puppet for server provisioning.

'Orchestration' means arranging or coordinating multiple systems. It's also used to mean "running the same tasks on a bunch of servers at once, but not necessarily all of them."

http://codefol.io/posts/deployment-versus-provisioning-versus-orchestration

TECHTOWN

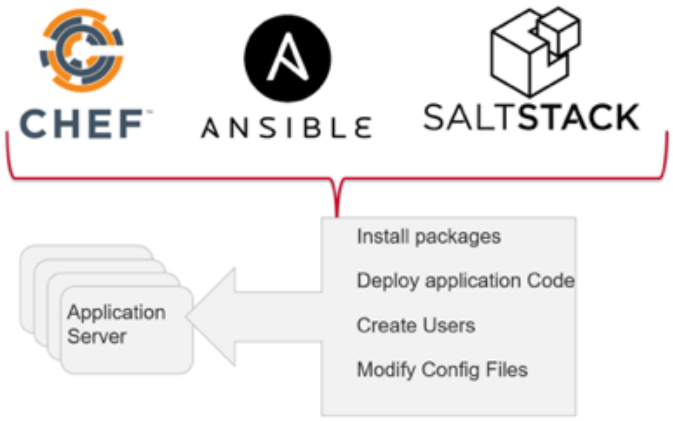
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

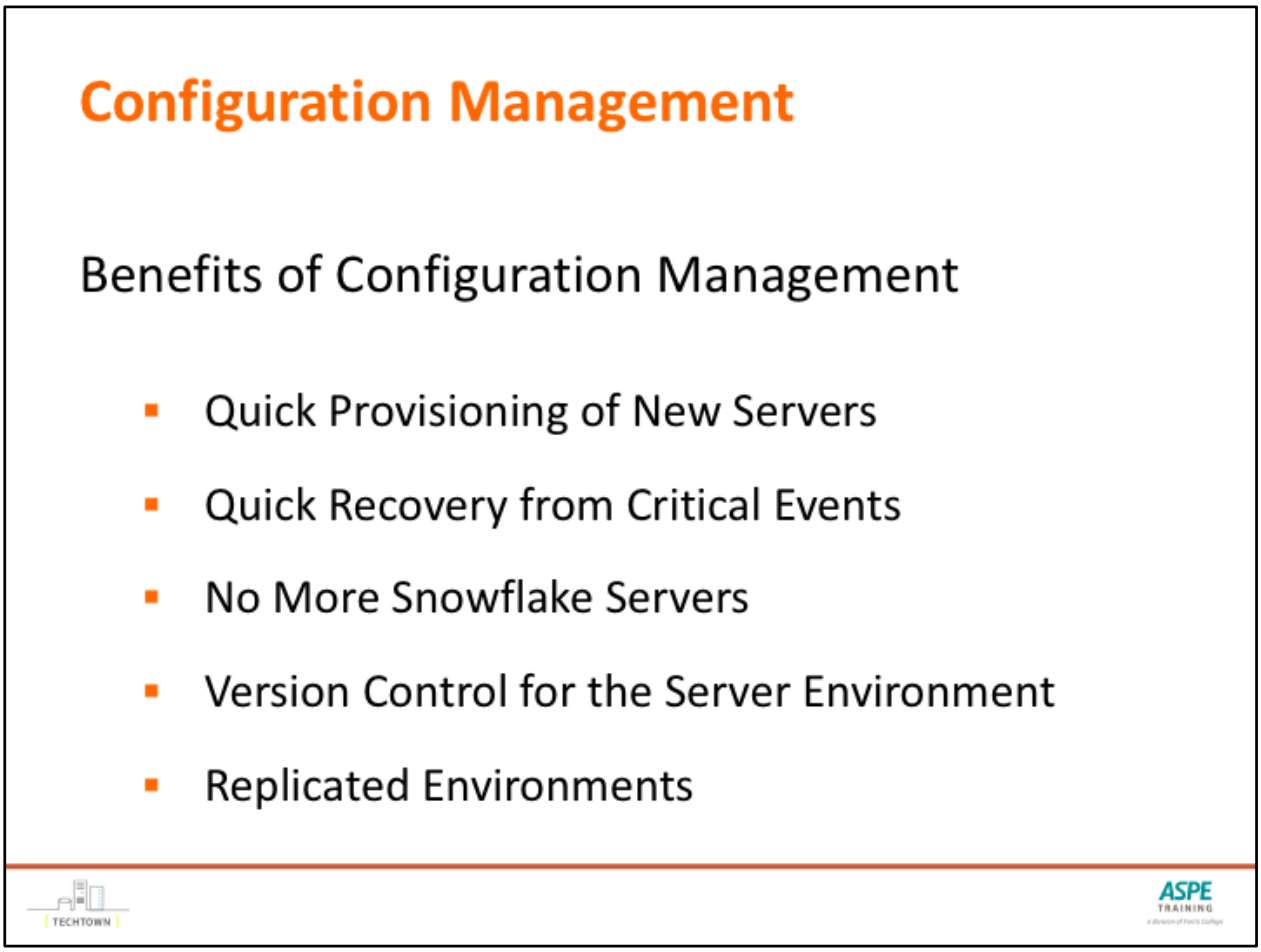
Configuration Management

- Process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.
- Usually does not ORDER automation steps
- Usually acts on existing infrastructure



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



There are a number of configuration management tools available in the market. Puppet, Ansible, Chef and Salt are popular choices. Although each tool will have its own characteristics and work in slightly different ways, they are all driven by the same purpose: to make sure the system's state matches the state described by your provisioning scripts.

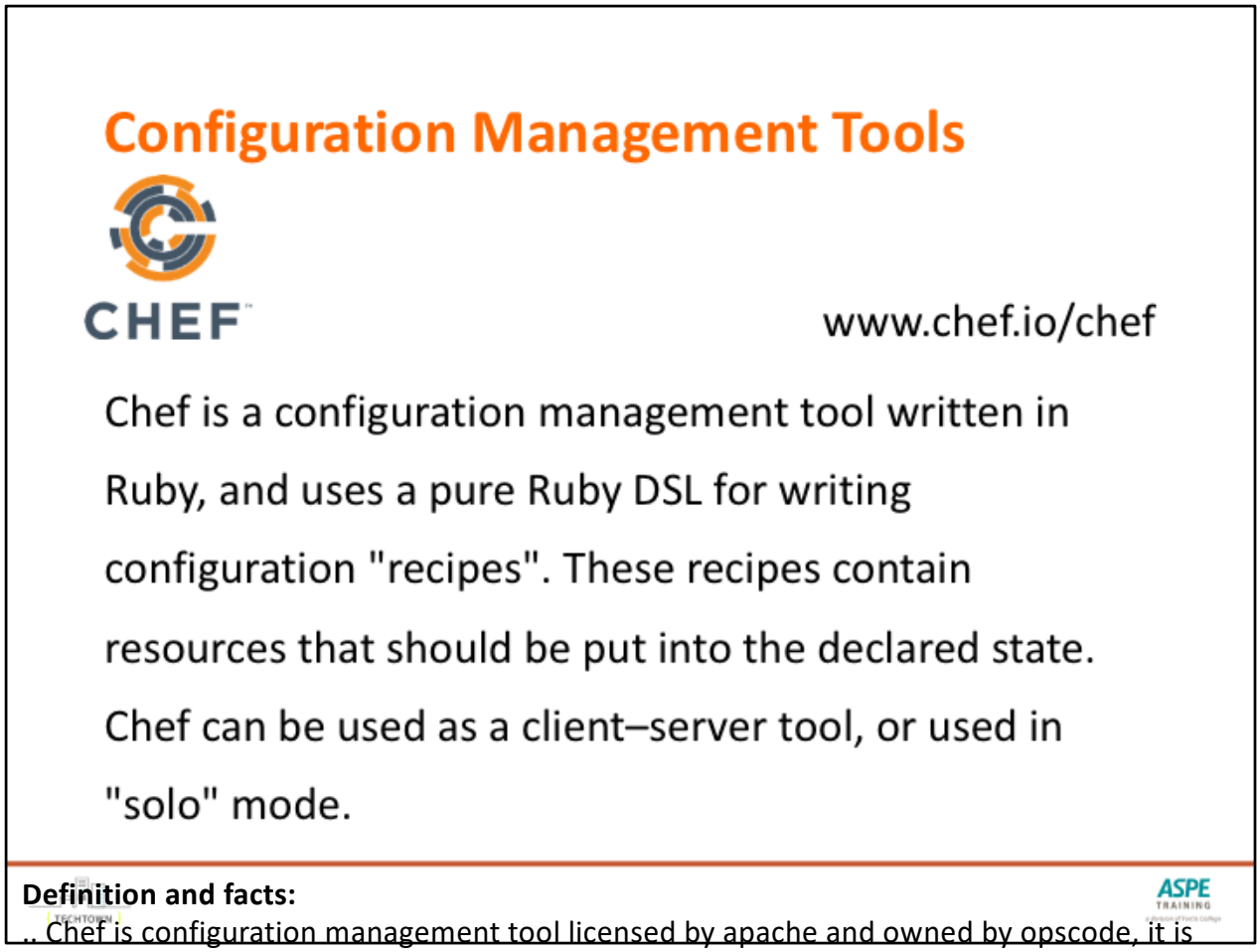
Benefits of Configuration Management for Servers
Although the use of configuration management typically requires more initial planning and effort than manual system administration, all but the simplest of server infrastructures will be improved by the benefits that it provides.

To name a few:

Quick Provisioning of New Servers
Whenever a new server needs to be deployed, a configuration management tool can automate most, if not all, of the provisioning process for you. Automation makes provisioning much quicker and more efficient because it allows tedious tasks to be performed faster and more accurately than any human could. Even with proper and thorough documentation, manually deploying a web server, for instance, could take hours compared to a few minutes with configuration management/automation.

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Definition and facts:

.. Chef is configuration management tool licensed by apache and owned by opscode, it is written in Ruby and Erlang.

.. Chef use a client-server architecture and is designed for programmers

.. Chef has three components – Node, workstation, server

> Workstation to develop and test your code (recipes)

> Server – is the master that manage the cluster

> Node – slaves that contains actual recipes and are part of environment (dev, test, prod)

.. Tools of chef – recipes, cookbooks, kitchen, knife

> Recipes contains resources and actions, defined to put the system into defined state. E.g. install package, create file, etc

> Cookbooks is just like a project and used to manage multiple recipes

> Kitchen is used to create virtual machines using docker or vagrant. And helps in testing recipes in different environments seamlessly

> Knife is used to interact with server and run anything in multiple nodes at once.

.. Solr search – Knife also comes with solr search. So, you can write one liner for things as complex as “run a piece of code where machine hardware matches this regex and/or ip address starts with this, etc.”

Pros:

> It’s flexible cloud infrastructure automation framework allows users to install apps to bare metal VMs and cloud containers.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

> Can scale to thousands of nodes easily

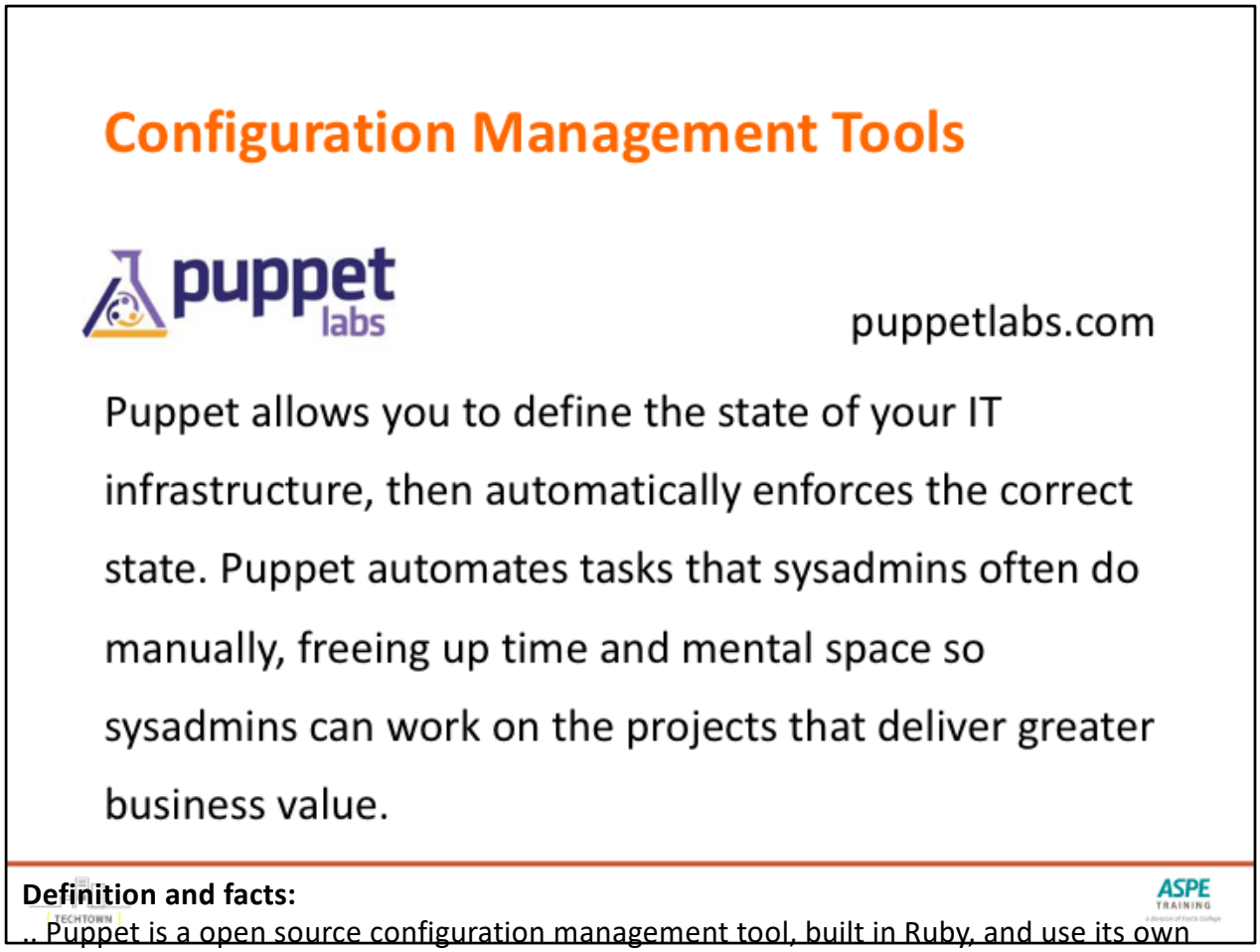
Cons:
> requires a steep learning curve
> If not a programmer, it may be difficult at start
> push-pull model:
pull model – daemon running in each machine and periodically checks for new instruction from master. Chef/Puppet are pull models
push model – push changes from your local machine and is applied to all nodes in the cluster. Ansible is a push model

Clients:
Facebook, Splunk, J&J. In fact, our company instantbrains helped J&J to configure Chef in their environment.

Compare Chef and Puppet:
> Chef is for developers, Puppet is for admins.
> Chef seems to be made with creative developers in mind, while Puppet caters more to cautious system administrators.
> Chef wants to help you make things. Puppet wants to help you not make mistakes.
> Chef have initial learning curve, once you are over it, you get a lot more power and flexibility. You can bend configurations to your will much more easily.

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Definition and facts:

.. Puppet is a open source configuration management tool, built in Ruby, and use its own DSL domain specific language, and embedded ruby templates.

.. Puppet is mostly for automating tasks for sysadmins, task like configuring, provisioning, troubleshooting and maintaining server operations.

.. Puppet uses an agent/master architecture – Master control configuration info and agents manage nodes and request relevant info from masters.

Pros:

> Puppet has strong compliance automation and reporting tools

Cons:

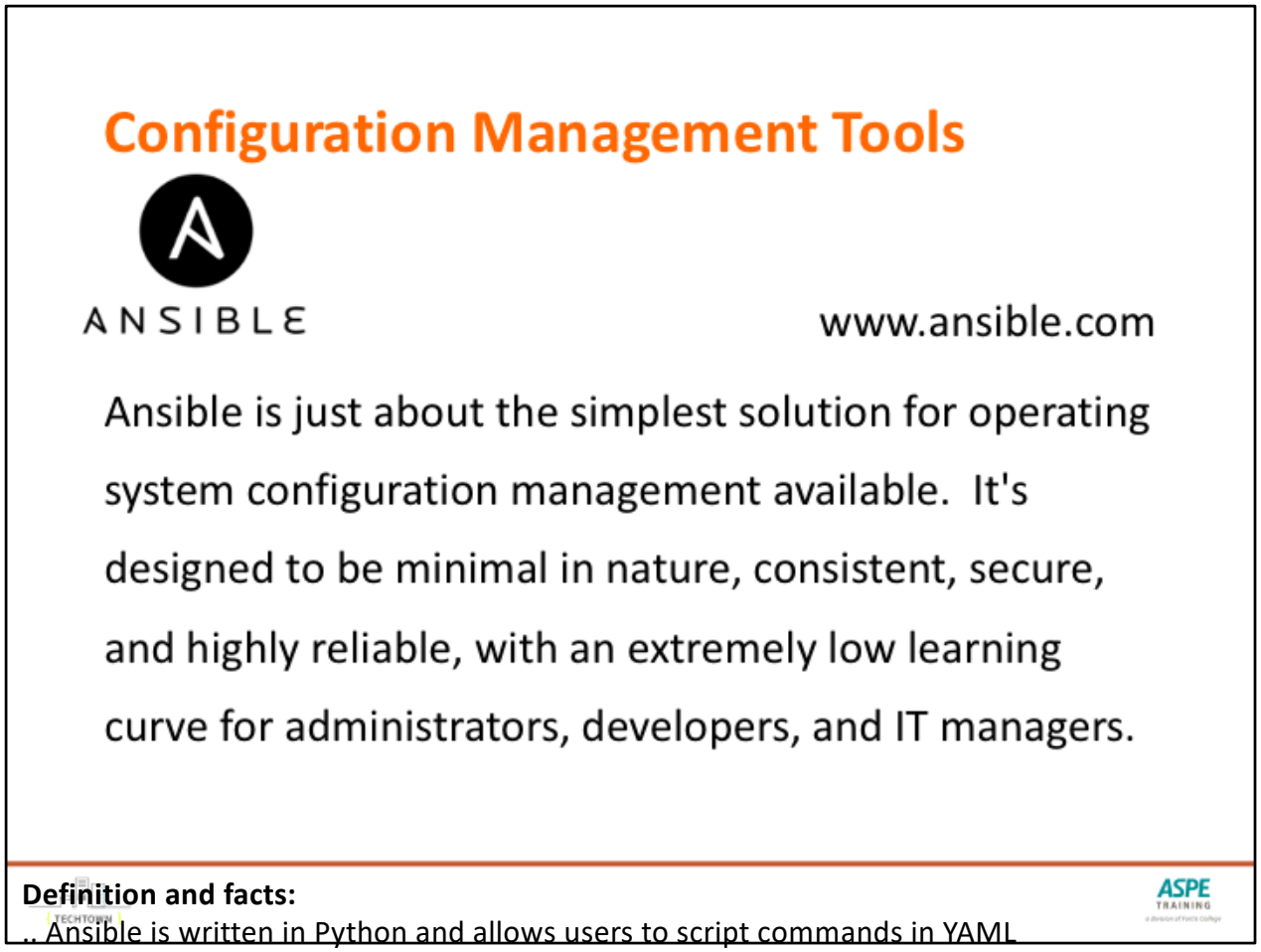
> Can be difficult for new users, and it is not the best solution available to scale deployments.

> Support is more focused toward Puppet DSL over pure Ruby versions. And due to this, it is less flexible than Chef

Clients:

Used by Salesforce, Sony, Google, Twitter, Nokia

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Definition and facts:

.. Ansible is written in Python and allows users to script commands in YAML

.. Ansible offers multiple push models to send command modules to nodes via SSH that are executed sequentially

.. Ansible doesn't require agents on every system, and modules can reside on any server.

Pros:

> Easy remote execution

> Suitable for environments designed to scale rapidly

> Shares facts between multiple servers, so they can query each other

> Syntax and workflow is fairly easy to learn for new users

> Supports both push and pull models

> Lack of master eliminates single point of failure.

Cons:

> Increased focus on orchestration (arranging or coordinating multiple systems) over configuration management.

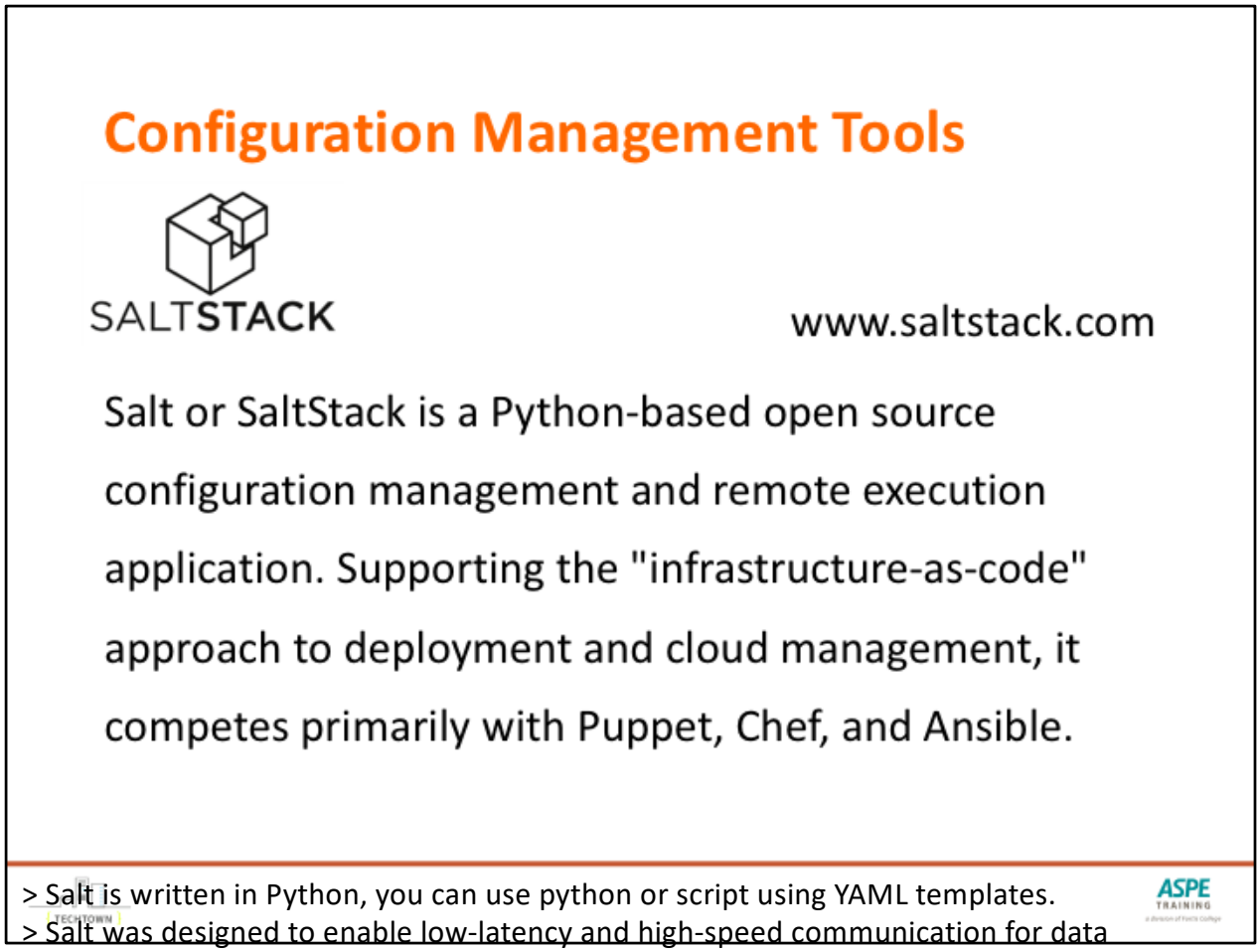
> SSH communication slows down in scaled environments.

> Underdeveloped GUI with limited features.

> The platform is new and not entirely mature as compared to Puppet and Chef.

Clients: Used by Samsung, Exxon mobile, coca cola

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



> Salt is written in Python, you can use python or script using YAML templates.
> Salt was designed to enable low-latency and high-speed communication for data collection and remote execution in sysadmin environments.
> Salt supports master-agent, de-centralized and non-master models
> Uses the push model for executing commands via SSH protocol, similar to Ansible
> Salt allows parallel execution of multiple commands
> A single master can manage multiple masters.

Pros:

> Effective for high scalability and resilient environments
> Easy and straightforward usage past the initial installation and setup.
> Python offers a low learning curve for developers.

Cons:

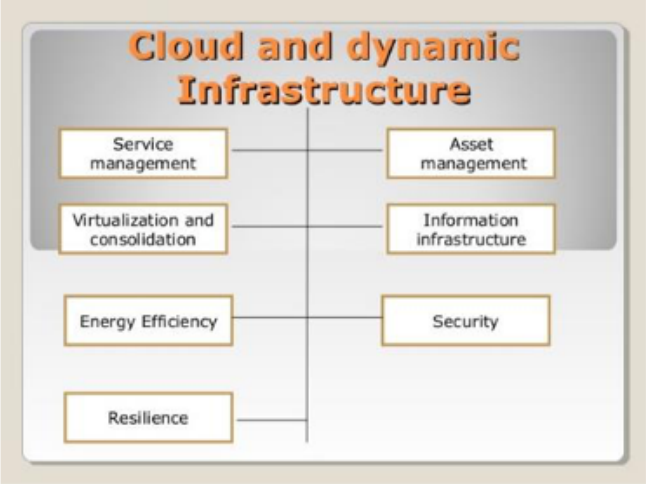
> Installation process may not be smooth for new users.
> Documentation is not well managed, and is challenging to review.
> Web UI offers limited capabilities and features.
> Not the best option for OSs other than Linux.
> The platform is new and not entirely mature as compared to Puppet and Chef.

Clients: Used by jobsprings, market share is pretty low, less than 2%

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Infrastructure in the Cloud

- How Infrastructure in the Cloud Can Be Utilized in the Enterprise

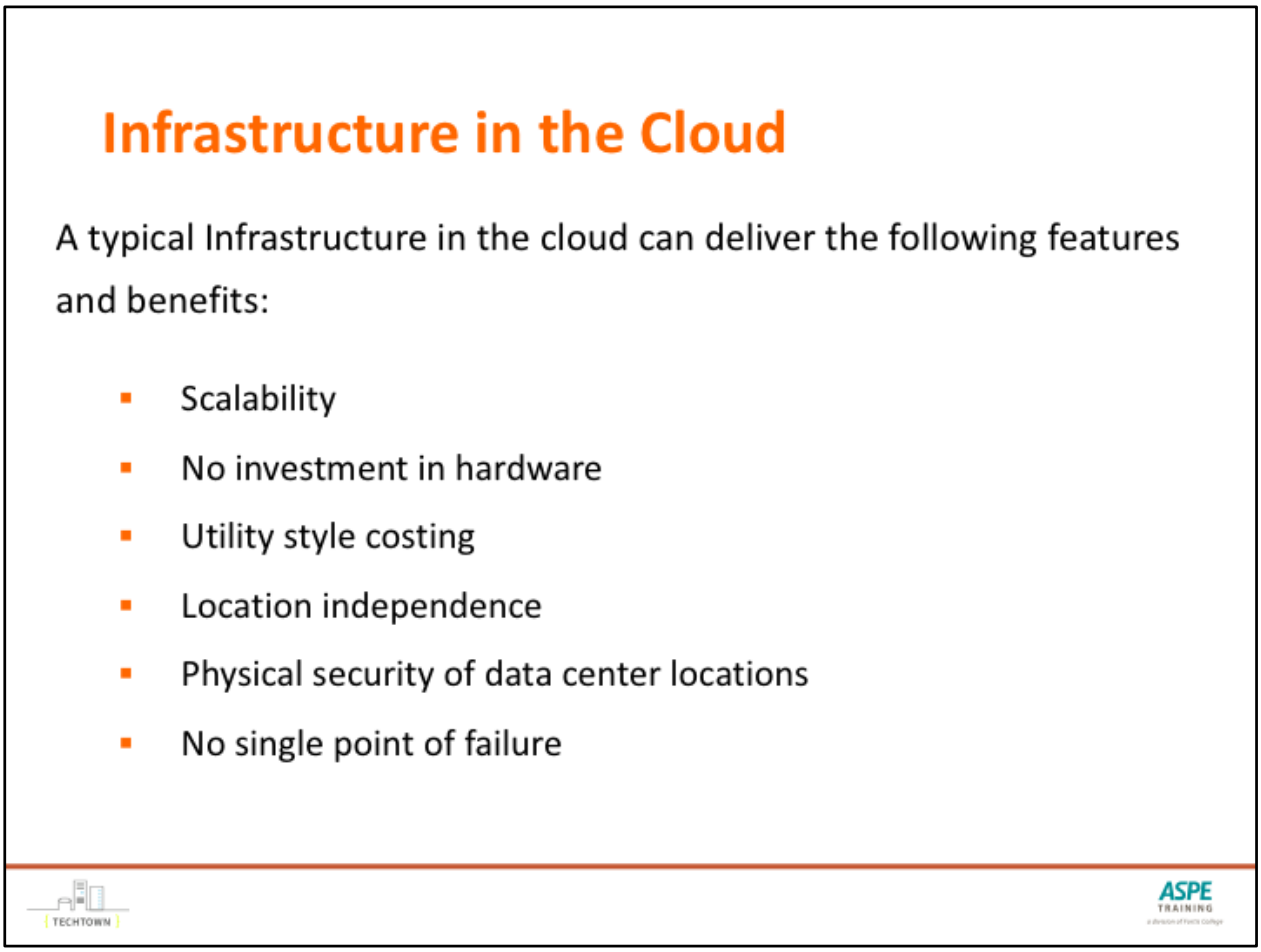


Enterprise infrastructure; by internal business networks, such as private clouds and virtual local area networks, which utilize pooled server and networking resources and in which a business can store their data and run the applications they need to operate day-to-day. Expanding businesses can scale their infrastructure in accordance with their growth whilst private clouds (accessible only by the business itself) can protect the storage and transfer of the sensitive data that some businesses are required to handle.

Cloud hosting; the hosting of websites on virtual servers which are founded upon pooled resources from underlying physical servers. A website hosted in the cloud, for example, can benefit from the redundancy provided by a vast network of physical servers and on-demand scalability to deal with unexpected demands placed on the website.

Virtual Data Centers (VDC); a virtualized network of interconnected virtual servers which can be used to offer enhanced cloud hosting capabilities, enterprise IT infrastructure or to integrate all of these operations within either a private or public cloud implementation.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



A typical Infrastructure in the cloud can deliver the following features and benefits:

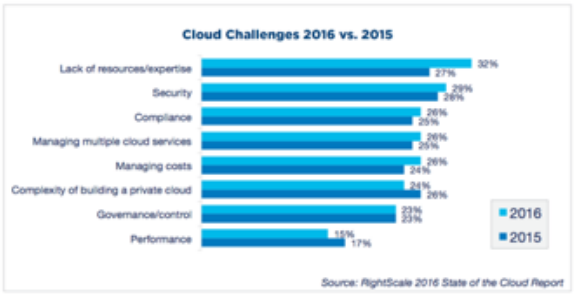
- Scalability; resource is available as and when the client needs it and, therefore, there are no delays in expanding capacity or the wastage of unused capacity
- No investment in hardware; the underlying physical hardware that supports an IaaS service is set up and maintained by the cloud provider, saving the time and cost of doing so on the client side
- Utility style costing; the service can be accessed on demand and the client only pays for the resource that they actually use
- Location independence; the service can usually be accessed from any location as long as there is an internet connection and the security protocol of the cloud allows it
- Physical security of data center locations; services available through a public cloud, or private clouds hosted externally with the cloud provider, benefit from the physical security afforded to the servers which are hosted within a data center
- No single point of failure; if one server or network switch, for example, were to fail, the broader service would be unaffected due to the remaining multitude of hardware resources and redundancy configurations. For many services if one entire data center were to go offline, never mind one server, the IaaS service could still run successfully.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Public Cloud Challenges

Top Public Cloud Challenges

- Lack of Resources/Expertise
- Security
- Compliance
- Managing Multiple Cloud Providers
- Managing Costs
- Complexity of Building a Private/Hybrid Cloud



Source: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2016-state-cloud-survey

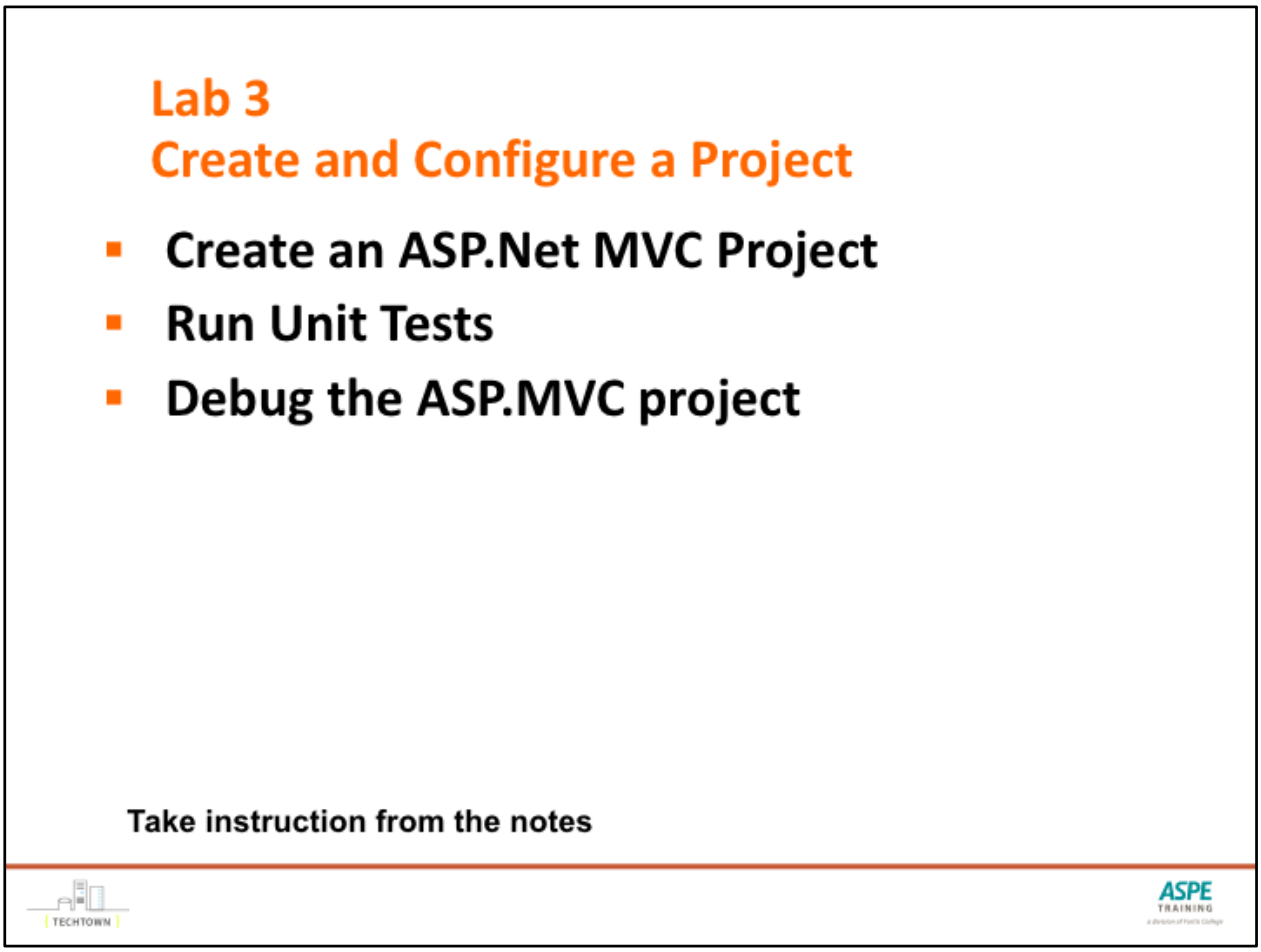
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise
Identify Deployment Improvements

- Update your Value Stream Map based to reflect using Infrastructure as Code
  - Estimate new Lead Times and %CAs
- Identify how Infrastructure as Code improves your Deployment process

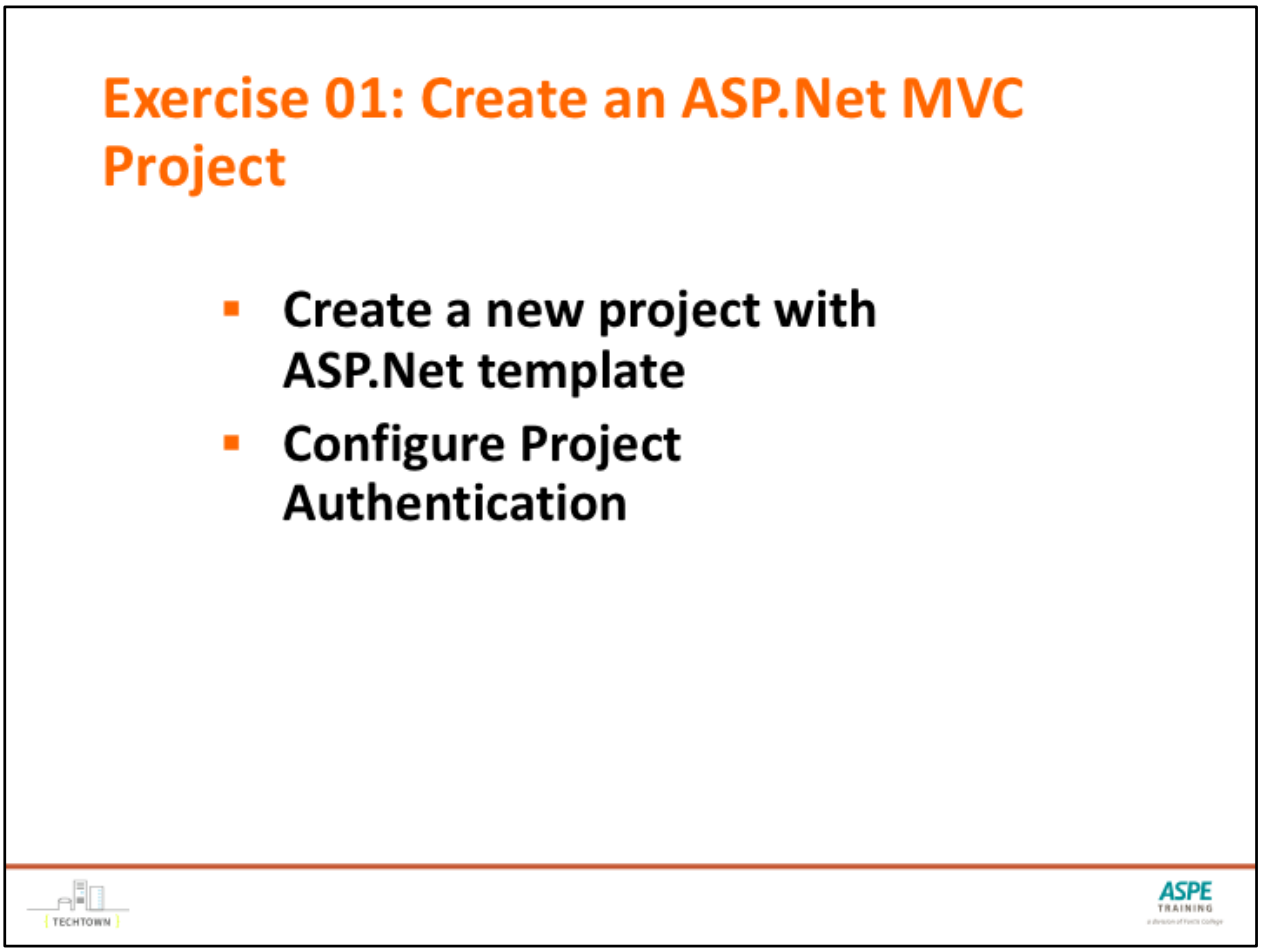
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In this lab we will configure an ASP.Net MVC Project and it's associated Database that we will later build and with Continuous Integration and eventually deploy to Microsoft Azure when our build completes successfully.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 1:

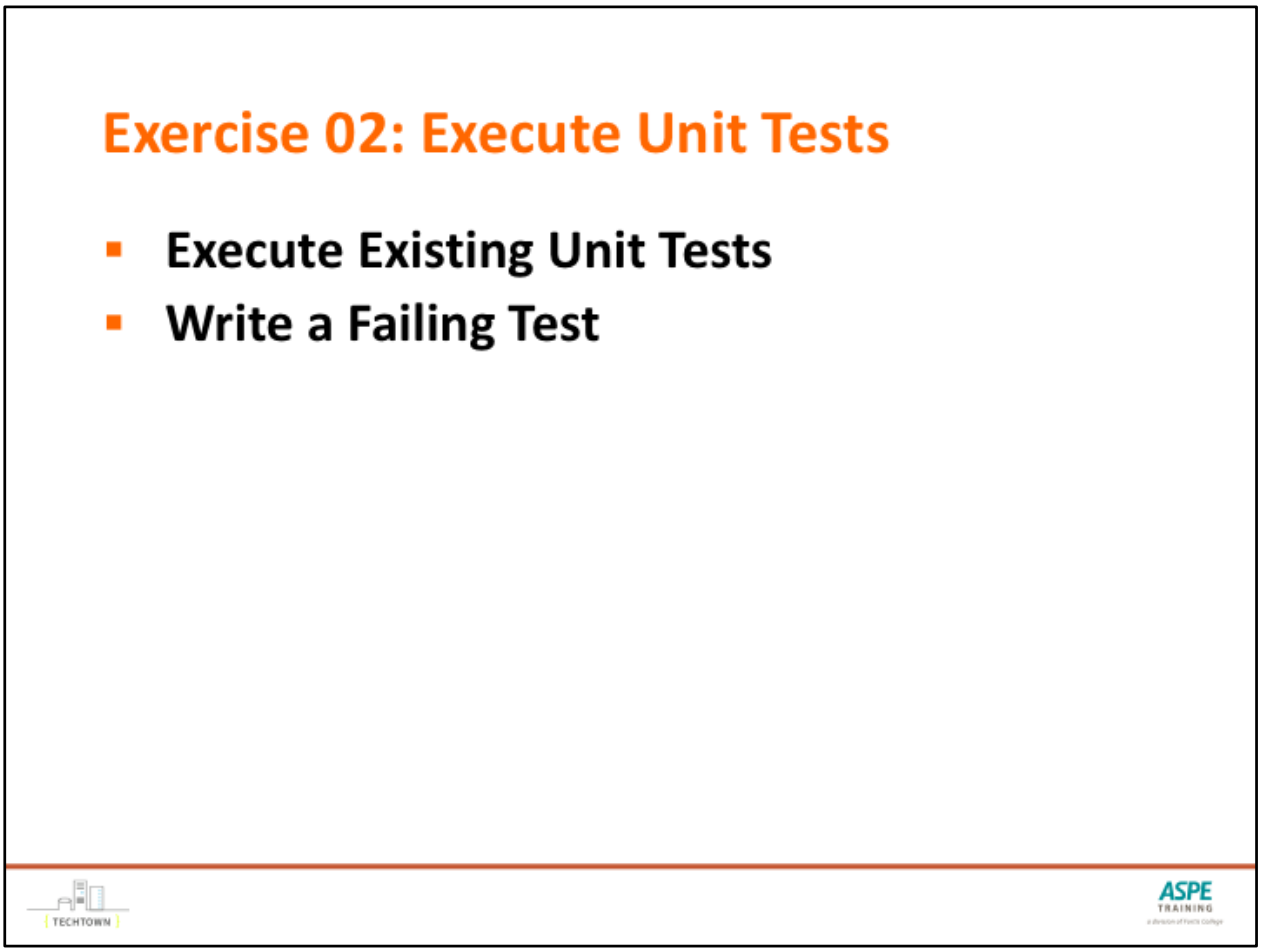
In this exercise, we will create an **ASP.NET MVC** project in order to facilitate the creation of Unit Tests, and ensure that our changes are pushed from **Team Explorer** to **TFS.**

Create a new project with ASP.NET template

1. Open up a browser, and navigate to **http://DevOpsClassroom.VisualStudio.com**
2. Log in as **User**: DevOpsStudent@Outlook.com **Password**: JustM300.
3. Navigate to the project that you created in the lab demo (Service Reservations)
4. Click on the **Dashboards** tab, and choose the **Open in Visual Studio** option in the lower left corner (this will link the Team Explorer to the **Team Foundation Server.)**
5. If the dialogue box pops up, select **Visual Studio Version selector**.

Configure Project Authentication

1. Click on the **Team Explorer** tab in the lower left corner.
   Notice that your the **Team explorer** is connected to your **Service Reservation** project.
   (We won't be using any of the available options at this point, but this will save us the pain of doing it manually later.)
2. Navigate to **File > New > Project.**
3. When the New Project dialogue comes up, expand the **Visual C#** menu and select the **Web** folder
4. Choose the **ASP.NET Web Application (.Net Framework)** option
5. Name the project **Service Reservation WebUI.**
6. In the **Solution name** field, remove **WebUI** so it reads as **Service Reservation** only, and click **OK.**
7. When the next dialogue box pops up, choose the **MVC** option, and check the **Add unit tests** checkbox.
8. Click on the **Change Authentication** button, and select the **Individual user** accounts option by clicking the radial button.
9. Make sure you have the **MVC** checkbox & **Add unit tests** checked, and click **OK**



Exercise 2: Execute Unit Tests

In this Exercise, we will run our existing Unit Tests, and explore the importance of writing a test that we know will fail.

Review Controller Methods and Test

1. In the **Solution Explorer**, locate the **Service Reservations WebUI** project and expand the **Controllers** folder
2. Select the **HomeController.cs**
   Notice that when you select **HomeController, in the code window** you can see a couple of functions – one for the **Index**, which returns the home page view.
   You can also see a function for the About page, which sets the **Viewbag.Message** to "**Your application description page**".
   Finally, you can see a function for the contact page
3. In the **Solution Explorer**, locate the **Service Reservations WebUI.Tests** project and expand the **Controllers** folder
4. Select the **HomeControllerTests.cs**
   Notice that when you select **HomeController, in the code window** you can see a couple of functions – one for the **Index**, which returns the home page view.
   You can also see a function for the about page, which verifies that the **Viewbag.Message** is set to "**Your application description page**".
   Finally, you can see a function for the contact page

Execute Existing Unit Tests

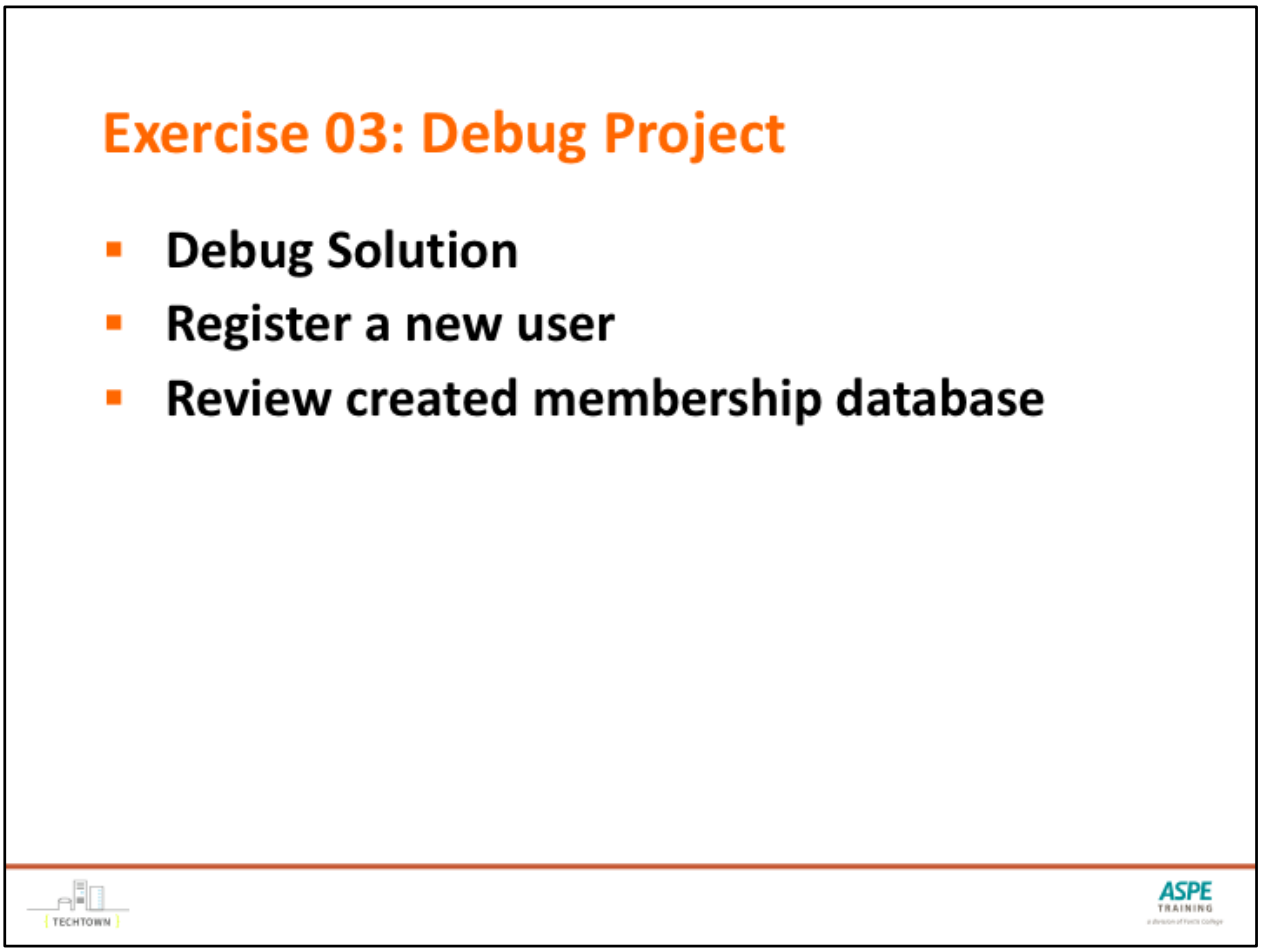
1. In the **Test** menu Select **Run > All Tests**
2. When the **Test Explorer** appears you should see the Test Results indicating that 3 of 3 Tests have passed
   Note: If the **Test Explorer** does not appear, from the **Test** menu select **Test > Window > Test Explorer**

Write a Failing Test

1. In the **Solution Explorer**, locate the **Service Reservations WebUI.Tests** project and expand the **Controllers** folder
2. Select the **HomeControllerTest.cs**
3. Edit the About Test Method and change the **Viewbag.Message** property from "**Your application description page**". to "**Service Reservation description page**".

Execute Existing Unit Tests

1. In the **Test** menu Select **Run > All Tests**
2. When the **Test Explorer** appears you should see the Test Results indicating that 2 of 3 Tests have passed and one has failed
   (Note: If the **Test Explorer** does not appear, from the **Test** menu select **Test > Window > Test Explorer)**



Exercise 3: Debug Project

In this exercise, we will debug our solution, register a new user so we can ensure that the membership database has been created.

Debug Solution

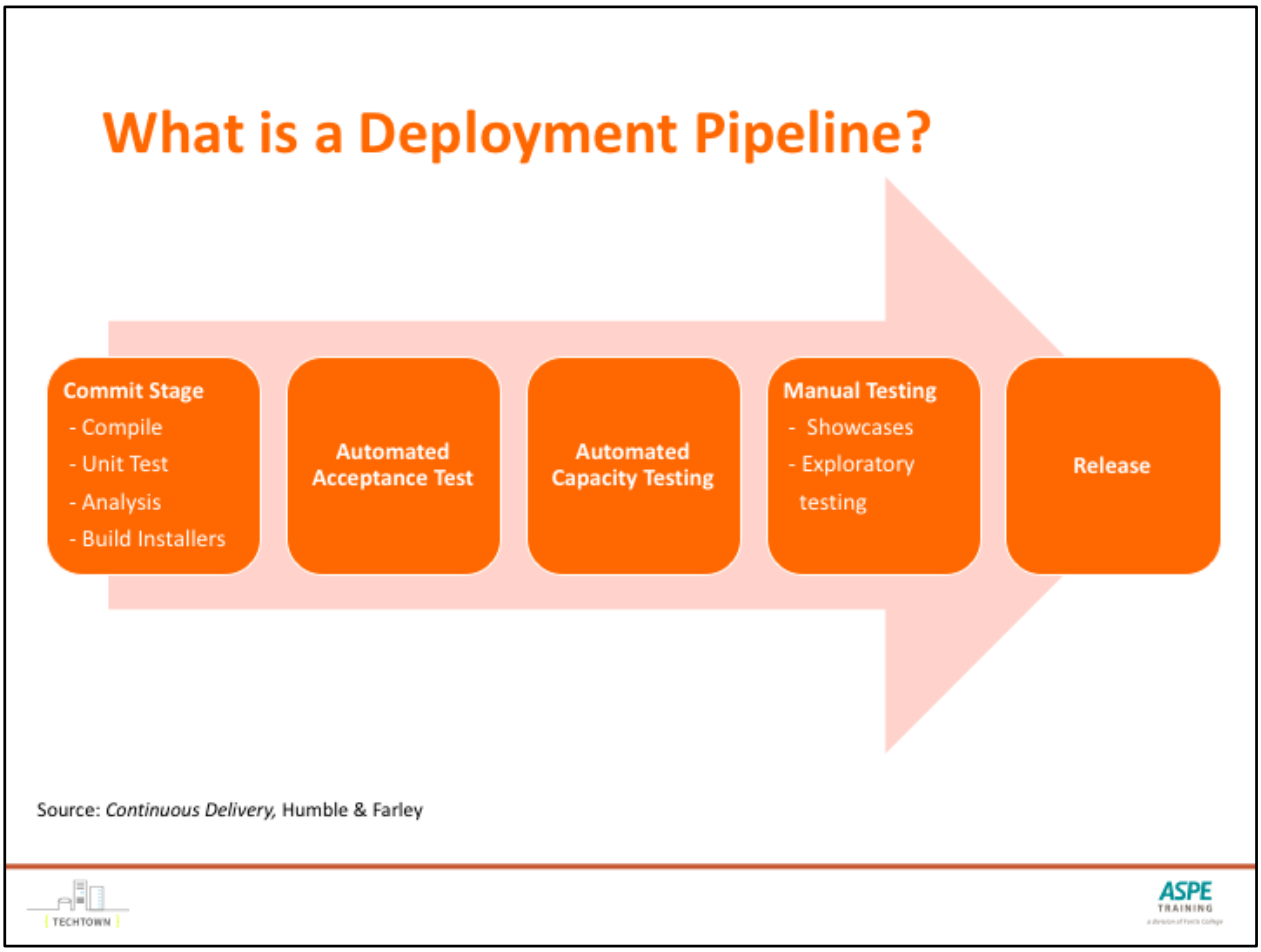
1. Click on the debug (play button) that reads **Google Chrome** to debug your project.
2. After the application compiles, and the Chrome browser appears, click on the **About** tab to see your description on the about page. (The ViewBag.Message that was passed in the About method that we looked at earlier).
3. Click on the **Contact** tab, and you can see the contact page, and you can see the ViewBag.Message that was passed in the Contact method that we looked at earlier).

Register a new user

1. We want to register a new user, so we can ensure that the membership database is created.
2. Click on **Register**, and choose the appropriate e-mail to enter
3. In the **Password** field, enter the password of your choice
4. In the **Confirm password** field, enter the password again
5. Click the **Register button.** You should now see the login that you chose as a welcome in the upper right corner.
6. Close the browser window to stop the debugging

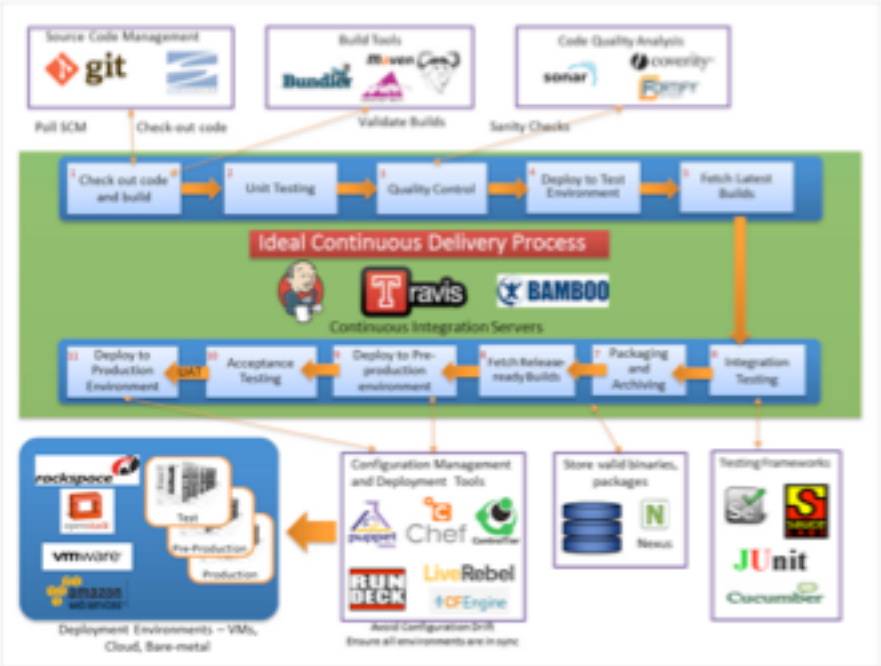
NOTES:

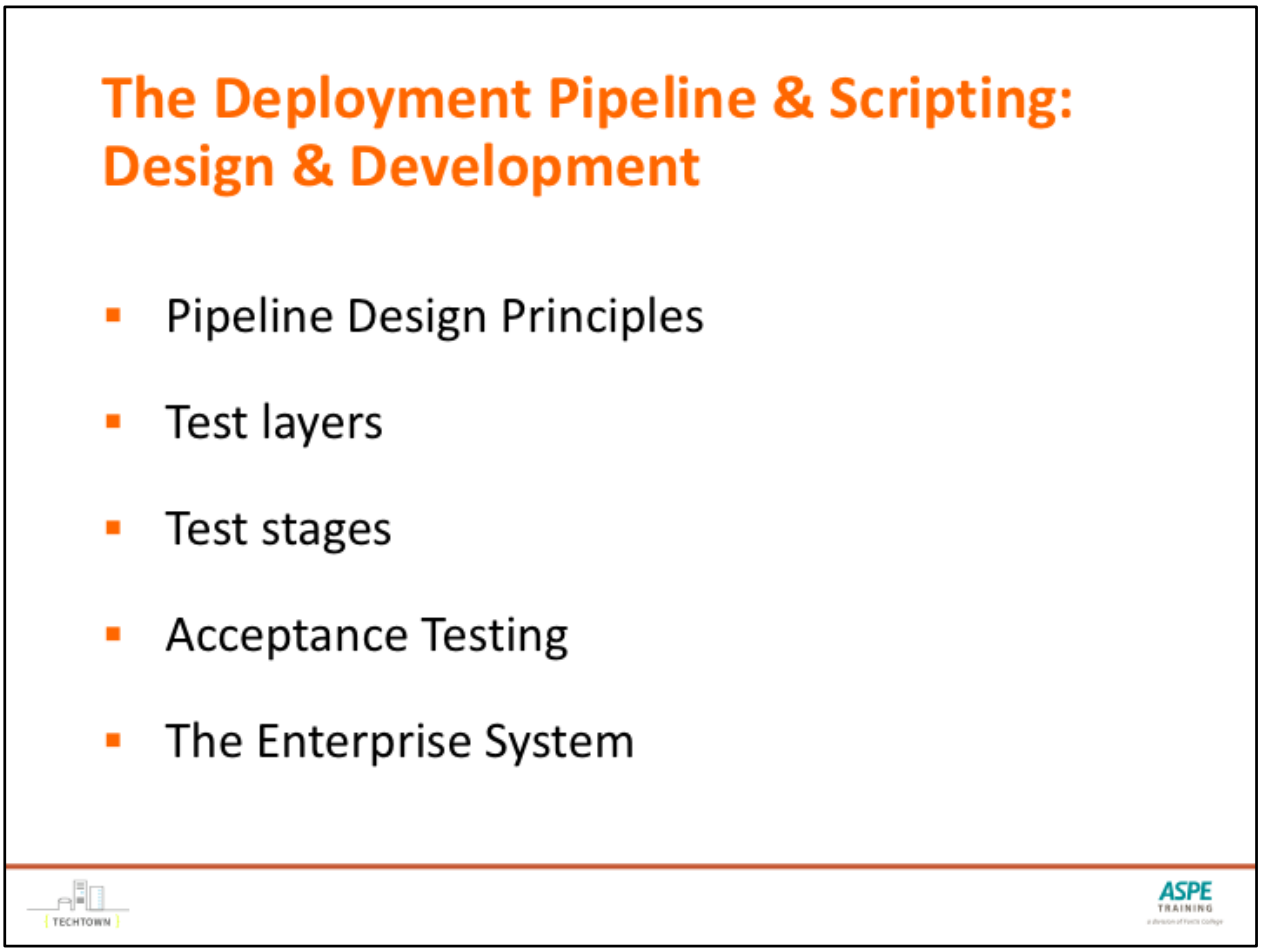
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In the ideal state, everything that makes up the complete software system is either a script (that is, a program) or an automated test. Everything that makes up the software system is versioned, and everything is incorporated into a continuous-delivery pipeline.

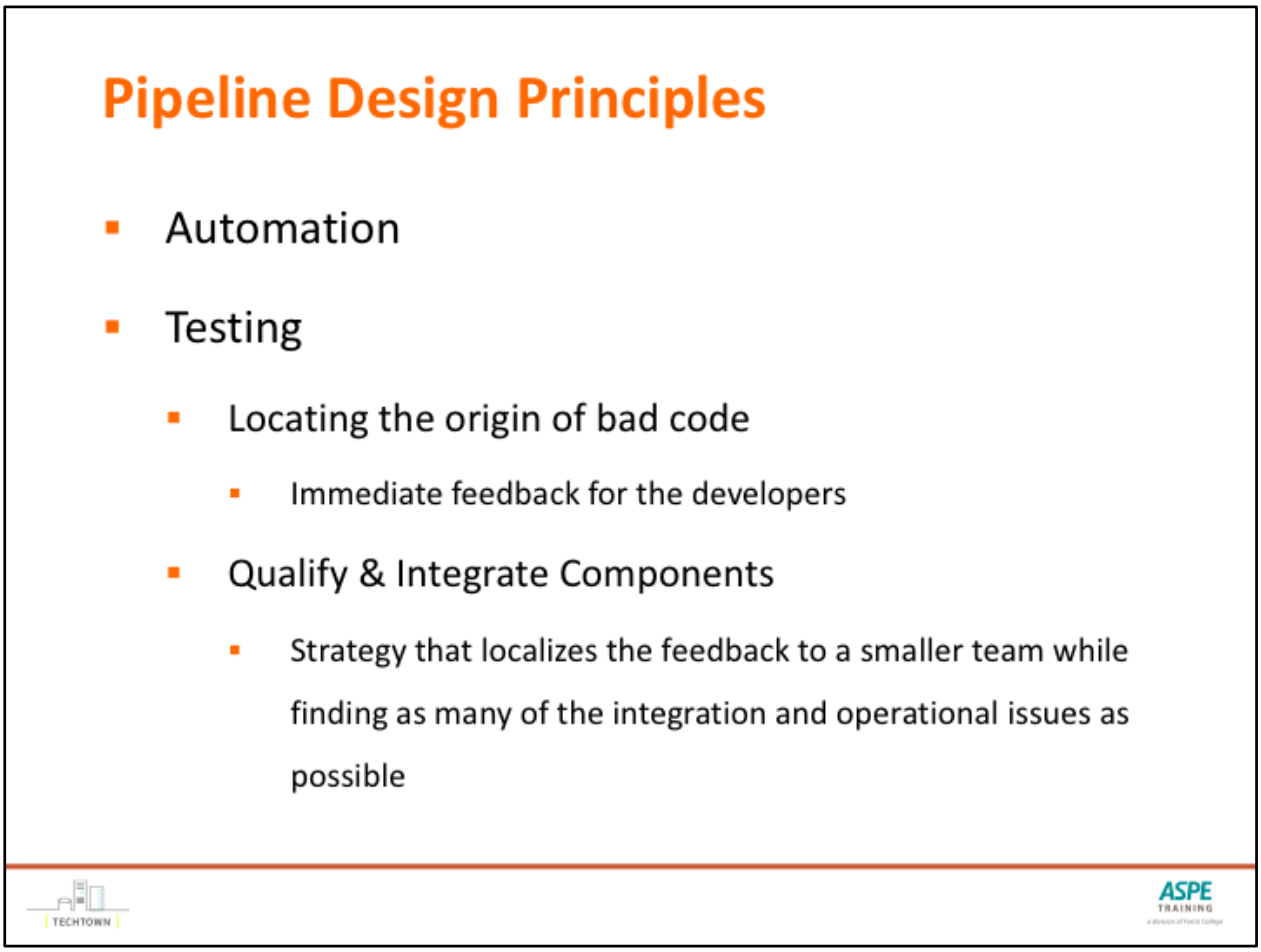
Here are some of the tools associated with a typical CD deployment pipeline:





Developers write better code when they have constructive feedback, and when that feedback is immediate, it is much easier for them to isolate and identify defects. In this chapter, we will dig in to some of the processes which help to provide consistent maintenance, ensure high code coverage, and prevent bad code from being added to the build.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



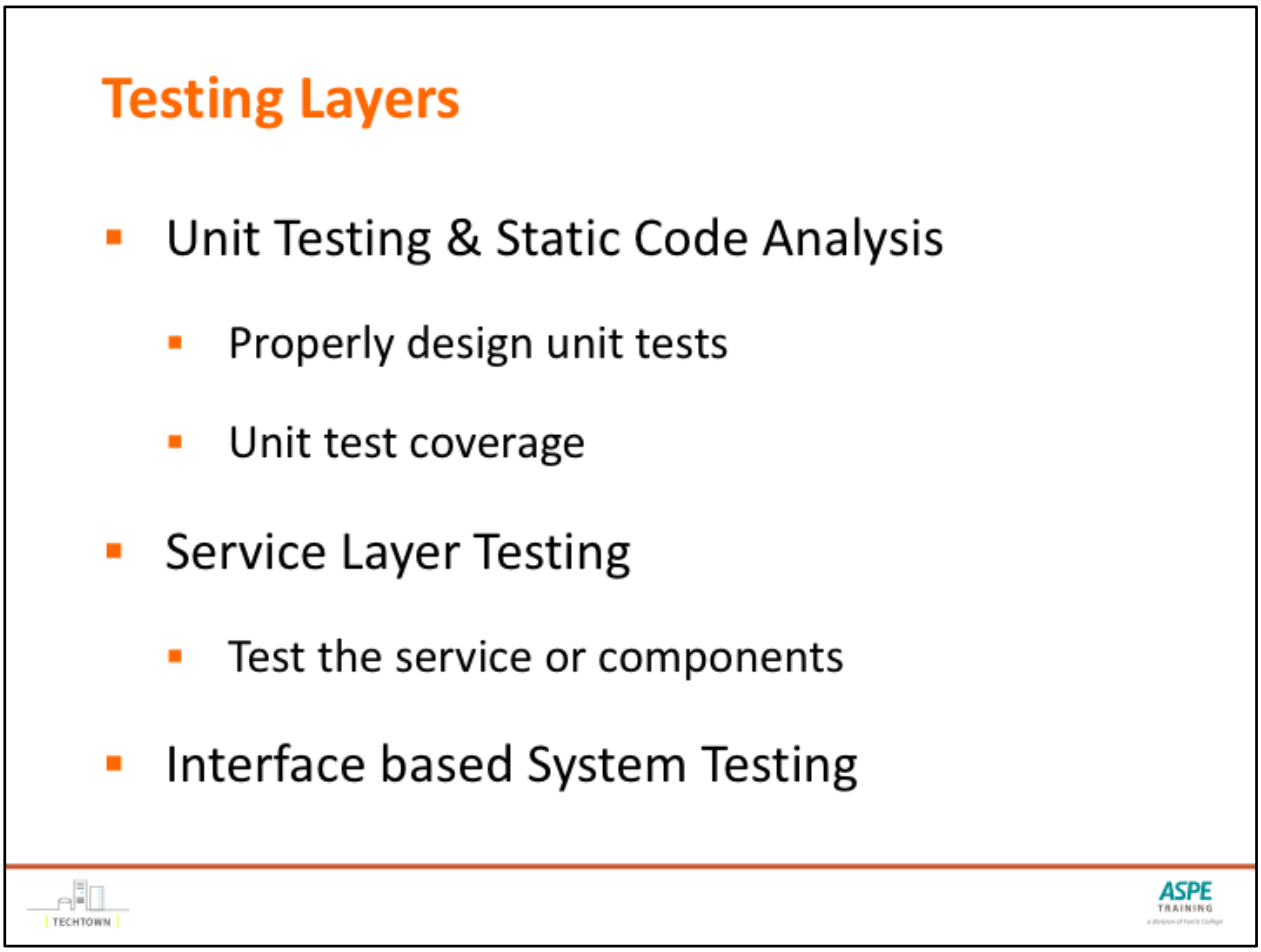
Locating the origin of bad code
For traditional large organizations, testing in layers and stages will help to detect the origin of bad code and when it was introduced. ***Points for discussion:***

- Immediate Feedback for Developers
- Reduced amount of Developers commit code
- Accurately Identify the source of bad code

Qualify & Integrate Components
To qualify and integrate components in a large organization, a strategy that localizes the feedback to the smallest number of teams while finding as many of the integration and operational issues as possible needs to be designed. ***Points for discussion:***

- Developers need code feedback on bad code
- Automated testing replicated on Desktop
- Verifying the fix before recommitting code

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Unit Testing & Static Code Analysis

- Properly designing Unit tests
- Catch defects as quickly as possible
- The advantages of Unit Testing

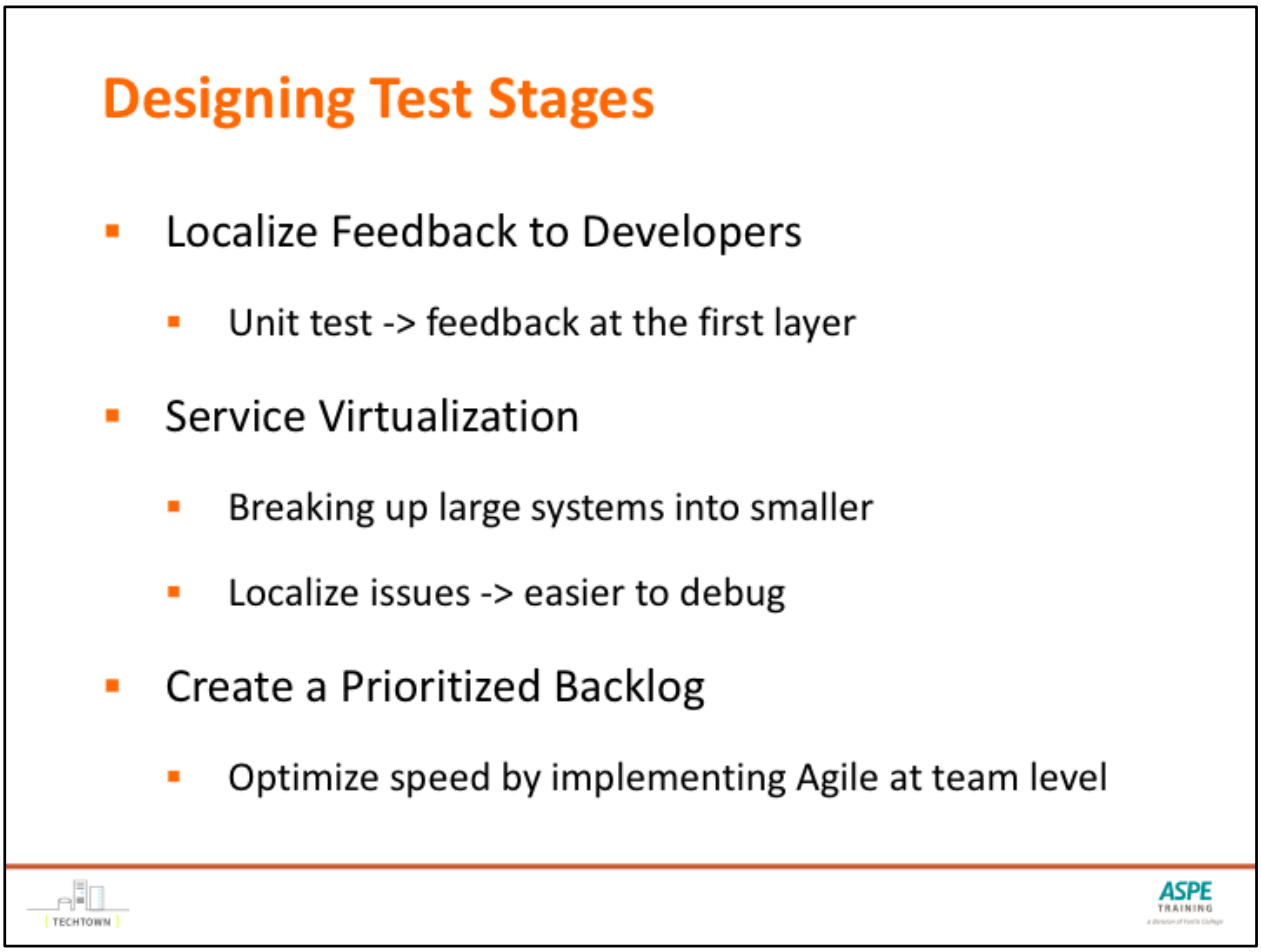
Service Layer Testing

- Can not find all defects with Unit testing
- Testing at the Service or Component Layer
- Qualify Code before final Integration

Interface based System Testing

- Using interface testing to find integration issues
- These tests run slowly, but are effective
- Must be run strategically

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Localize Feedback to Developers
It will be quicker and easier to detect code error sources when feedback for developers through automated testing is distributed to the smallest group possible. ***Points for discussion:***

- Design test strategy for smallest amount of developers possible
- Test as much as possible, as often as possible
- Detect integration & operational issues

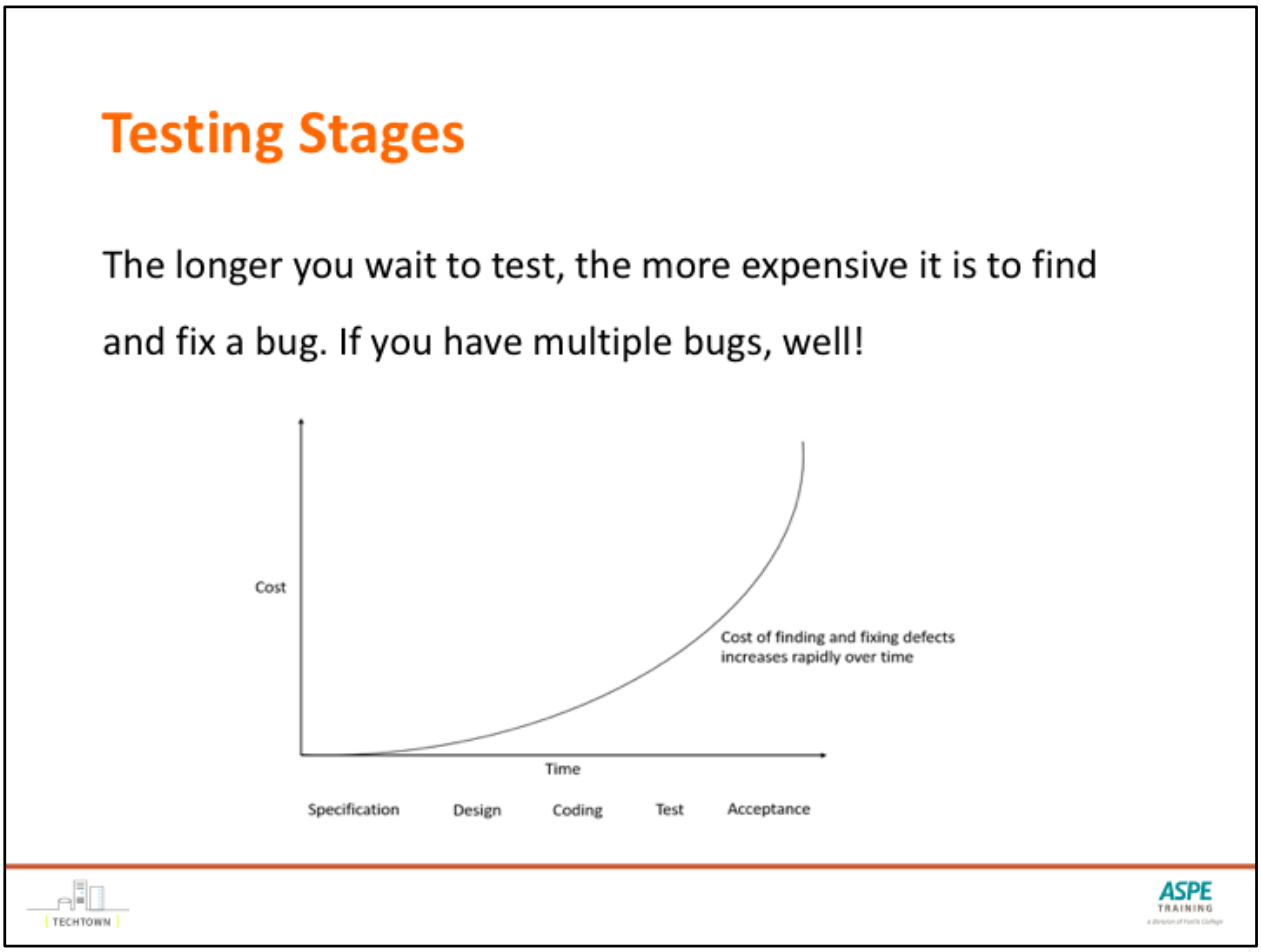
Service Virtualization
Service virtualization is a very effective tool for helping to break up a very large enterprise system into manageable pieces. ***Points for discussion:***

- Breaking up large systems into manageable pieces
- Localizing issues to the smallest amount of commits
- Use a subset for fast feedback & more extensive testing

Create a Prioritized Backlog
Creating prioritized backlog is a necessity for the transformation to Agile, and everyone should be involved on defining the tasks priorities. ***Points for discussion:***

- Implementing Agile at the Team level
- Optimize the Speed of Feedback
- Running Tests in smaller, less expensive environments

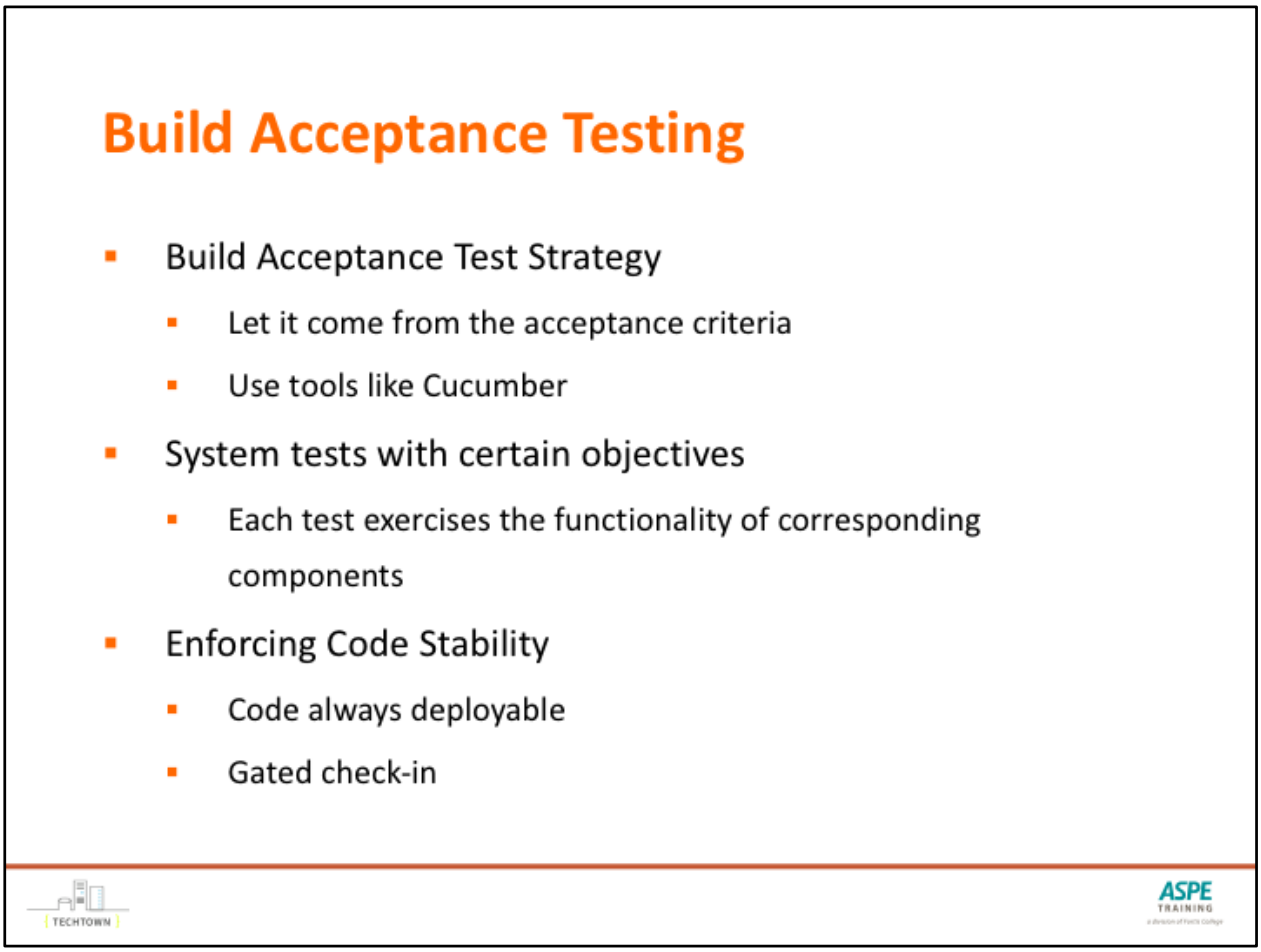
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



What is the cost of defects in software testing?

The cost of defects can be measured by the impact of the defects and when we find them. The earlier the defect is found, the less the cost of fixing it. For example, if error is found in the requirement specifications, then it is somewhat cheap to fix it. The correction to the requirement specification can be done and then it can be re-issued. In the same way when defect or error is found in the design then the design can be corrected and it can be re-issued. But if the error is not caught in the specifications and is not found till the user acceptance then the cost to fix those errors or defects will be way too expensive.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Build Acceptance Test Strategy
Design tests that are maintainable and can quickly localize failure to the right component, and break them into different stages to compensate for all possible defects. ***Points for discussion:***

- Breaking the System into groups
- Running System Tests
- Minimize delay in Feedback

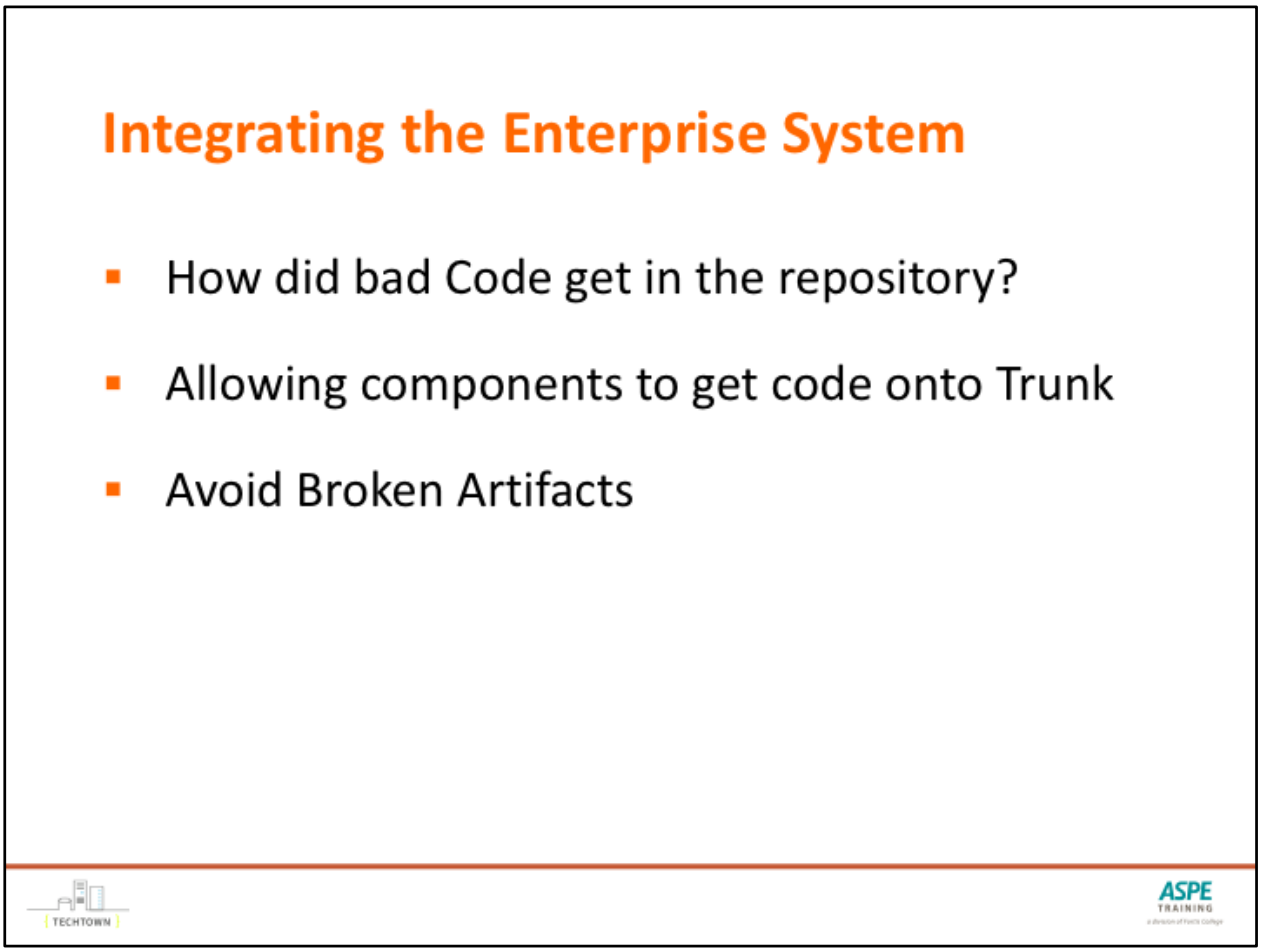
System tests with different objectives
Proper automation for acceptance tests dictates that each test only exercises the functionality of corresponding components. ***Points for discussion:***

- Effective Strategy for System Tests
- Apply the best coverage as quickly as possible

Enforcing Minimum Code Stability
The definition of minimum code stability should be enforced by gated check in and the design of the acceptance testing required for submission. ***Points for discussion:***

- Driving up quality of Code
- Fixing bad code is the top Priority

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



How did bad Code get in the repository?
When an error is detected in the acceptance test suite, it's an opportunity to detect where the error occurred, and bring the tests that let it through up to speed so they remain relevant. ***Points for discussion:***

- Balancing Code integration & Code Stability
- Automating Code Stability

Allowing components to get code onto Trunk
If faulty code makes it past tests into the repository, that means every developer on the project is working on a system that is fundamentally broken, thus the source of the error needs to be identified immediately. ***Points for discussion:***

- Enterprise Level Integration
- Providing visibility of Artifact Aging

Broken Artifact Train wrecks
When errors make it into the repository, Other developers on this team will continue to work on other tasks and their work will be tested by the same tests that are allowing bad code. ***Points for discussion:***

- Broken Code in the subsystem
- Taking ownership of fixing the issue
- Simplify the Deployment Pipeline

Deployment Pipeline and Scripting: Implementation

- What is a Deployment Pipeline?
- Committing Code
- Gated Acceptance Testing
- Automating Deployment
- Testing Stages
- Implementing a Deployment Pipeline
- Build Tools Overview
- Deployment Scripting
- Build Scripting
- Automating Tests

ASPE TRAINING

NOTES:

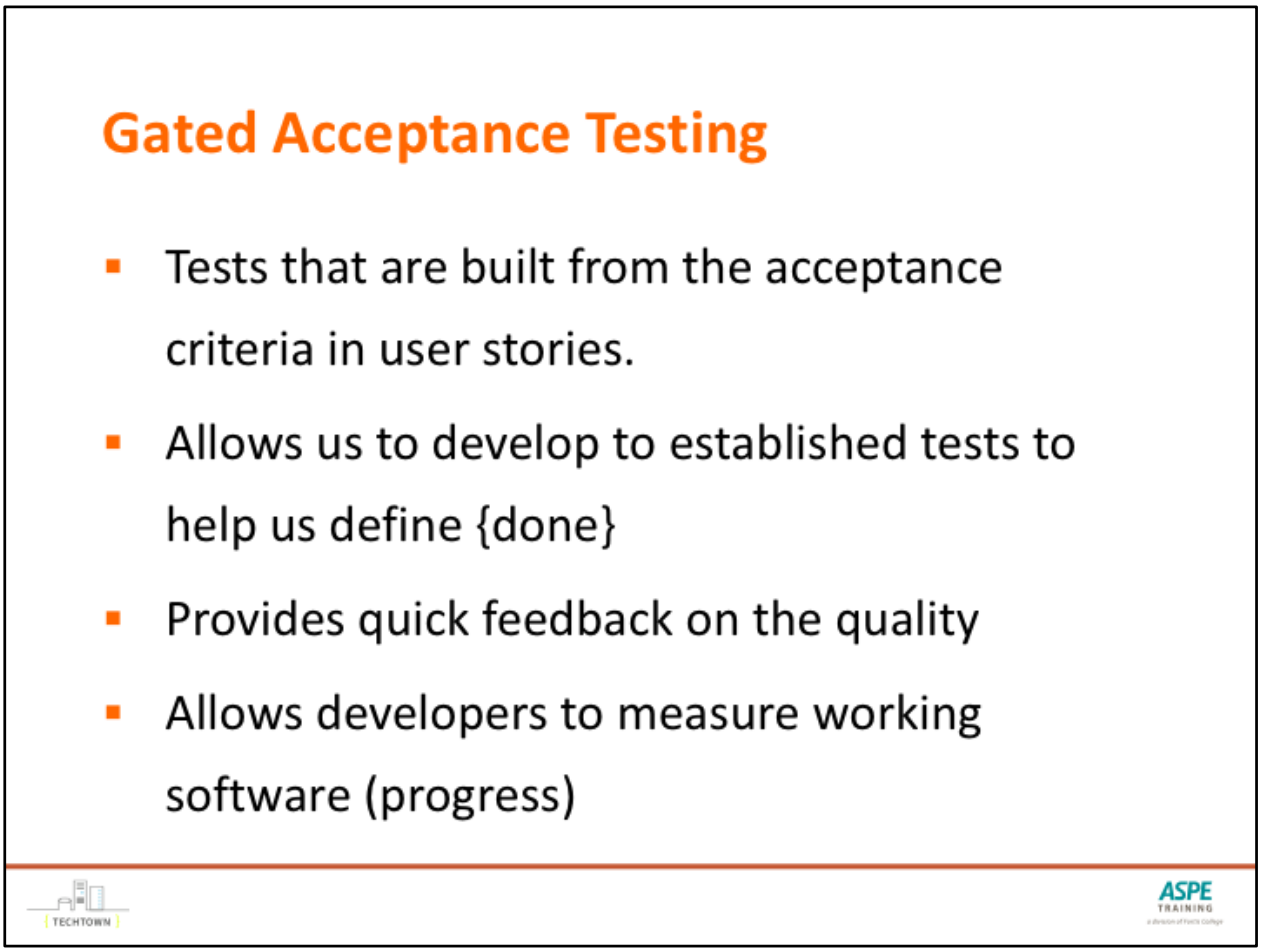
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Committing Code

- Commit Often
- Don't commit broken code
- Run Local Builds before commits
- Fail the build if any test fails
- Fix broken builds immediately
- Don't check out code from a broken build

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Keys to Good Acceptance Tests

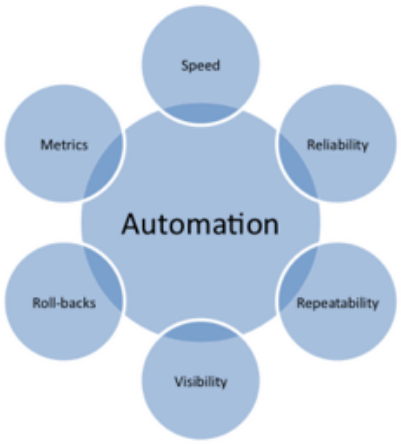
- Build a Testable Environment First
- Be Specific
- Use concrete examples
- Specify concrete behavior
- No ambiguity allowed
- Avoid Implementation Details

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Automating Deployment

Benefits

1. Allows anyone to deploy the application
2. New environments can be readily supported
3. Less overhead from deployments
4. More frequent deployments are possible
5. More scalable due to less errors



Automating deployments should be strongly considered. Even manually deploying a few servers as a one off can have unforeseen consequences if a server is missed. Automation is necessary to scale, not only for practical reasons such as one can only do so many machines manually but automation is better to ensure consistent environments, certify security vulnerabilities, and establish audit trails to assist in faster troubleshooting.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Build Tools Overview

There are plenty of tools available in the market, including

- make
- ant
- nant
- Msbuild
- Maven
- ivy
- gradle

Build Tools allows the automation of repeatable tasks by executing all the tasks in the correct, specific order and running each task accordingly. A build tool is used for building a new version of a program with all the specified requirements in correct manner.

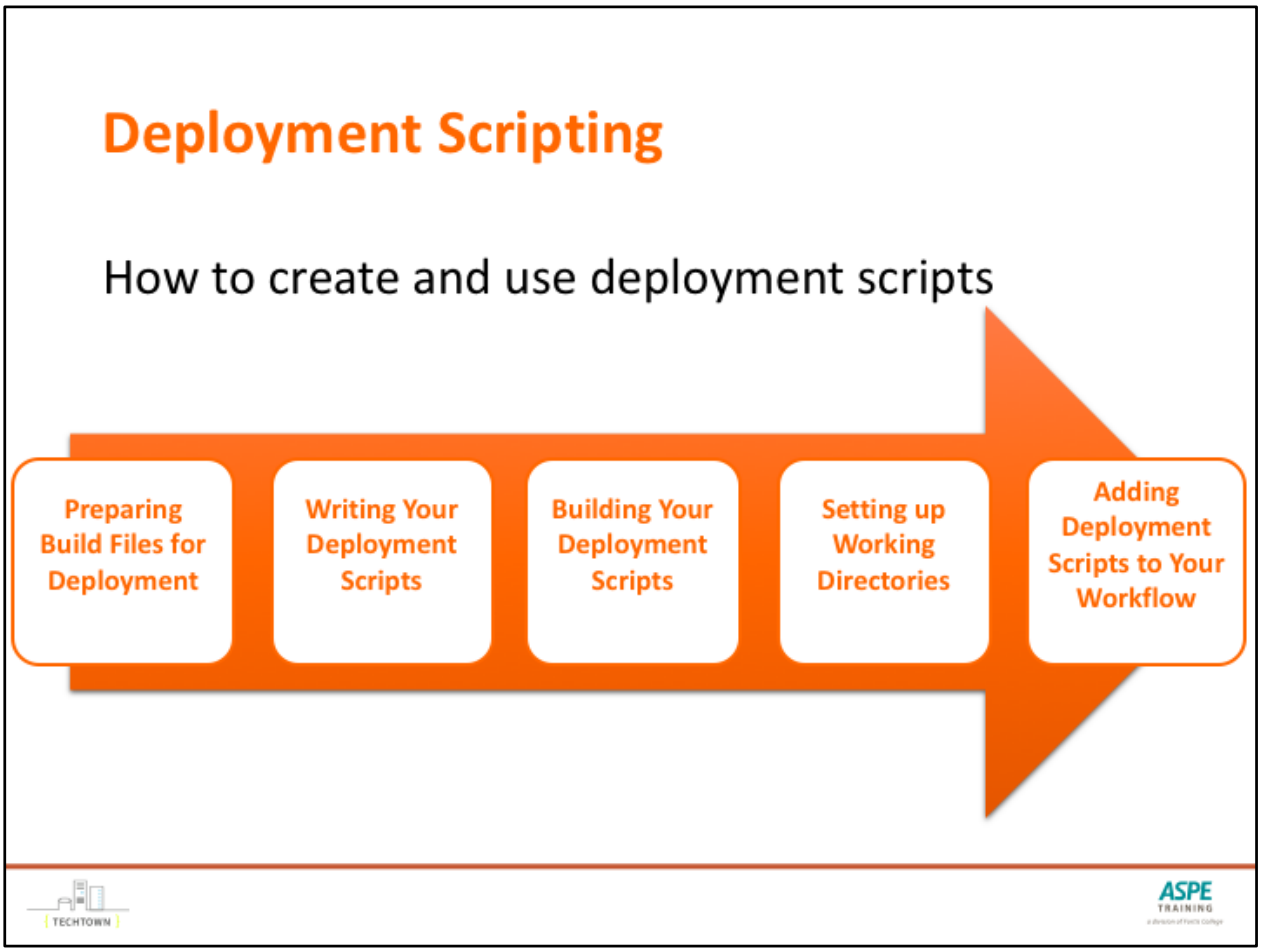
In the early days, people were using an Unix utility "**make**" which was an open source utility. **Make** uses the build file named as "Makefiles" which automatically builds the executable programs and libraries from source code. It also specifies how to derive the target program.

The revolution in build automation picked up a high speed when Apache introduced their open source build tool for java based application named as "**ant**". **Ant** supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications.

Ant uses XML as a build file in which users have to specify and write the order in which all the tasks are performed with desired outcome. By default the XML file is named as build.xml

Ant is better for controlling of build process and is a **perfect** fit for straight forward projects.
It has very simple structure thus allowing anyone to start using it without any special preparation and knowledge.

(continued on next page)

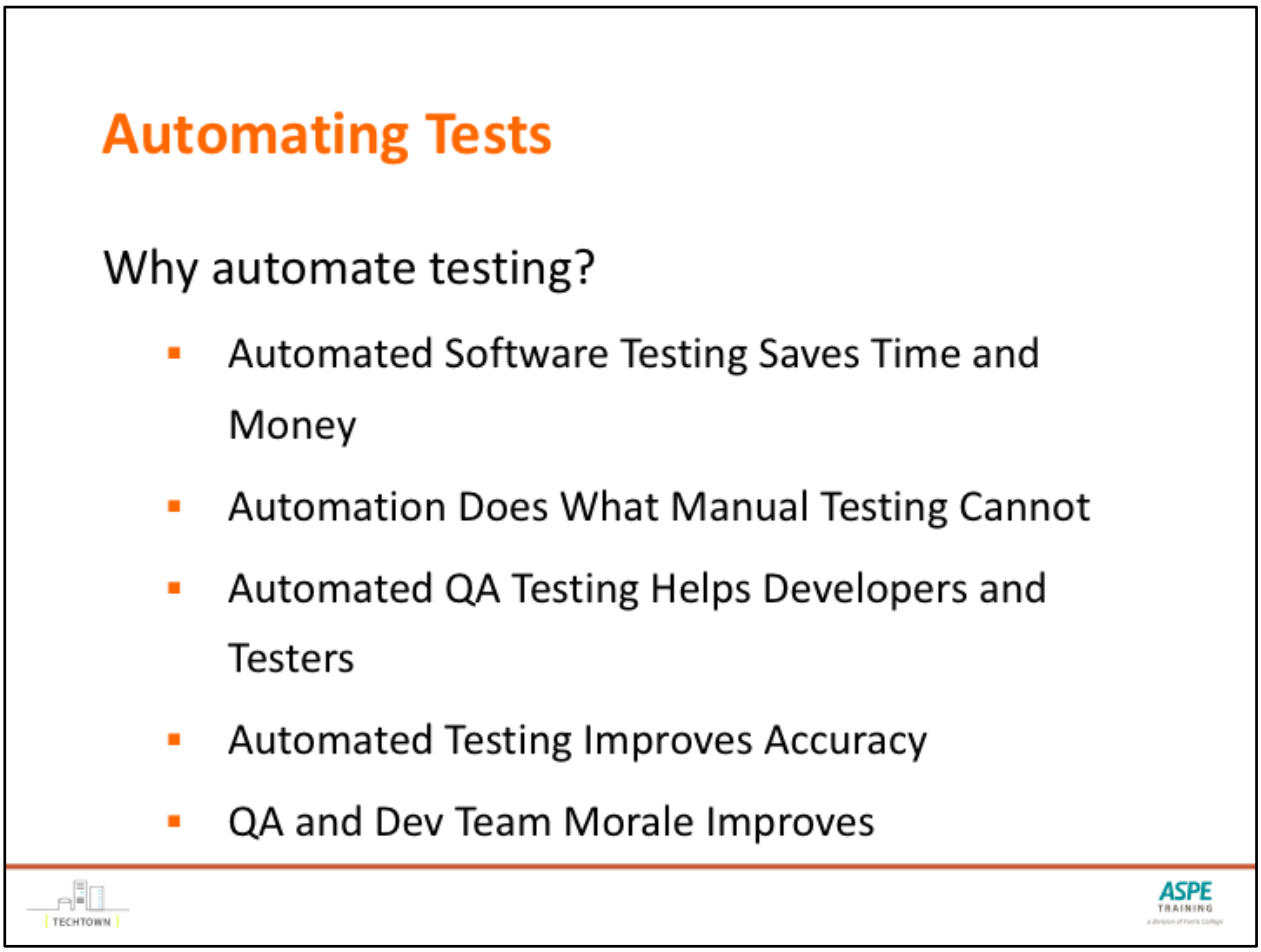


Writing your deployment scripts

These are the most common tasks carried out by deployment scripts:

- Get the build path on your build controller. You can send this to your deployment script as a command argument.
- Specify your deployment path.
- Create your deployment directory. You can also do this manually, instead of in your deployment script. If you are using pre-deployment environment snapshot with your workflow, you just need to create the directory on the virtual machines in your snapshot.
- Copy your deployment package from the build path to your deployment path.
- Run the deployment package in your deployment directory.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



There are several different types of software tests that can be automated.

- Unit Testing
- Functional Testing
- Regression Testing
- Black Box Testing
- Integration Testing
- Keyword Testing
- Data Driven Testing
- Smoke Testing

10 Tips you should read before automating your testing work

- So automate your testing procedure when you have lot of regression work.
- Automate your load testing work for creating virtual users to check load capacity of your application.
- Automate your testing work when your GUI is almost frozen but you have frequent functional changes.
- What are the Risks associated in Automation Testing?

What are the Risks associated in Automation Testing?

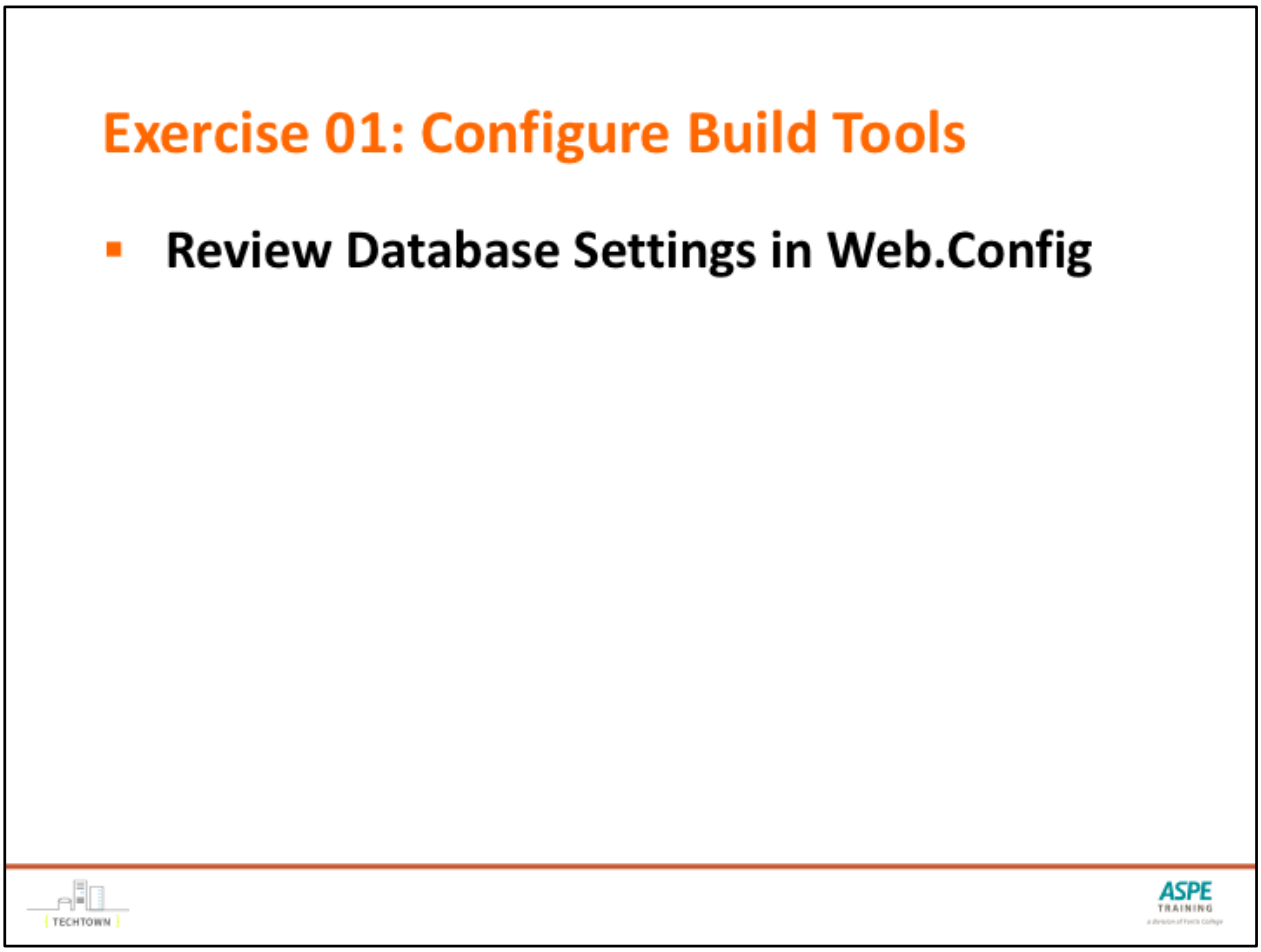
- Do you have skilled resources?
- Initial cost for automation is very high.
- Do not automate your UI if it is not fixed.
- Is your application stable enough to automate further testing work?
- Are you thinking of 100% automation?
- Do not automate tests that run once.
- Will your automation suite be having a long lifetime?

Lab: Automating Deployment

- Exercise 01: Configure Build Tools
- Exercise 02: Create Database Deployment Project
- Exercise 03: Check Solution into Source Control

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

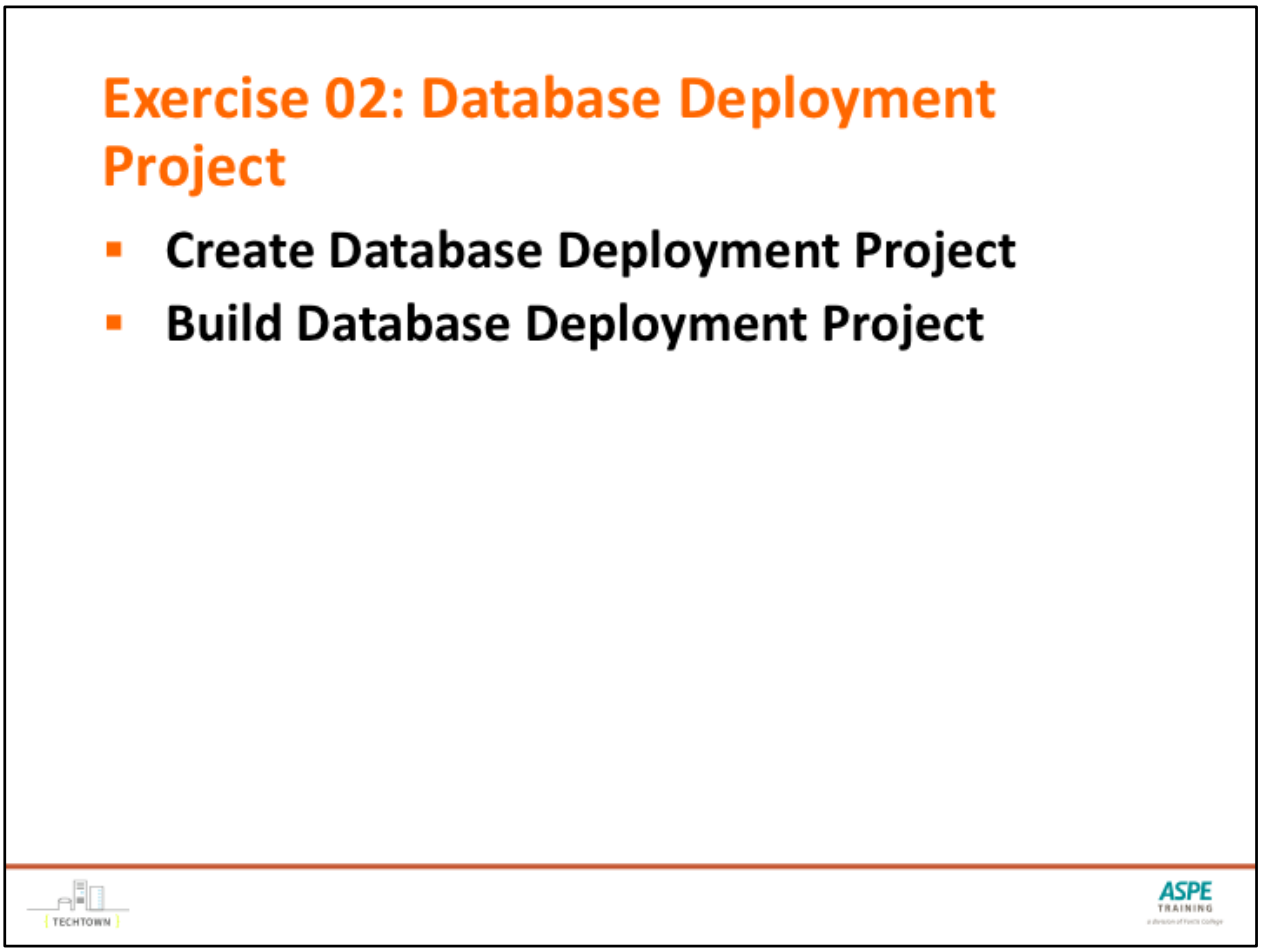


Exercise 1: Configure Build Tools
In this exercise, we will configure our build tools, and do a brief review of some of the database settings.

Review Database Settings in Web.Config
In Visual Studio we will review our solutions configuration and extract the database connection information

1. In our **Service Reservation Web UI** project, notice the default connection in the SQL Server Object Explorer. This default connection came from our projects **Web.config** file, which you can see in the **solution explorer**
2. Open the **Web.config** file. In the code window
   Notice we have connection strings, and a connection called **Defaultconnection**.
   Also notice the **Defaultconnection** settings:
   It has its data source set to **LocalDB MSSQLocalDB**;
   The Initial Catalog (aka the Dabasebase) is set to the project we created.
   Creating the file from our template automatically put the connection string in there.
   Debugging, and registering the user forced the creation of the database.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2: Database Deployment Project

In this exercise, we will Create a Deployment Project, And then create a build solution to make sure that what we've done has not introduced any defects to our production environment.

Create Database Deployment Project

In Visual Studio we will view the database created during user registration in the previous lab and create a database project from the existing database

1. In the **Server Explorer**, right click on the default connection, and select **Browse In SQL Server Object Explorer**
2. In the **Service Reservation WebUI Properties** window, check the path to your **WebUI** project, and follow it so we can verify the structure.
3. Follow the path all the way to the end, and you will see the **Service Reservation WebUI folder**, and the **Service Reservation WebUI.Tests** folder.
4. In the **SQL Server Object Explorer** locate your database, right click on it, and **create a new project.**
5. In the Import Database dialogue box, name the project **Service Reservations DB.**

(continued on next page)

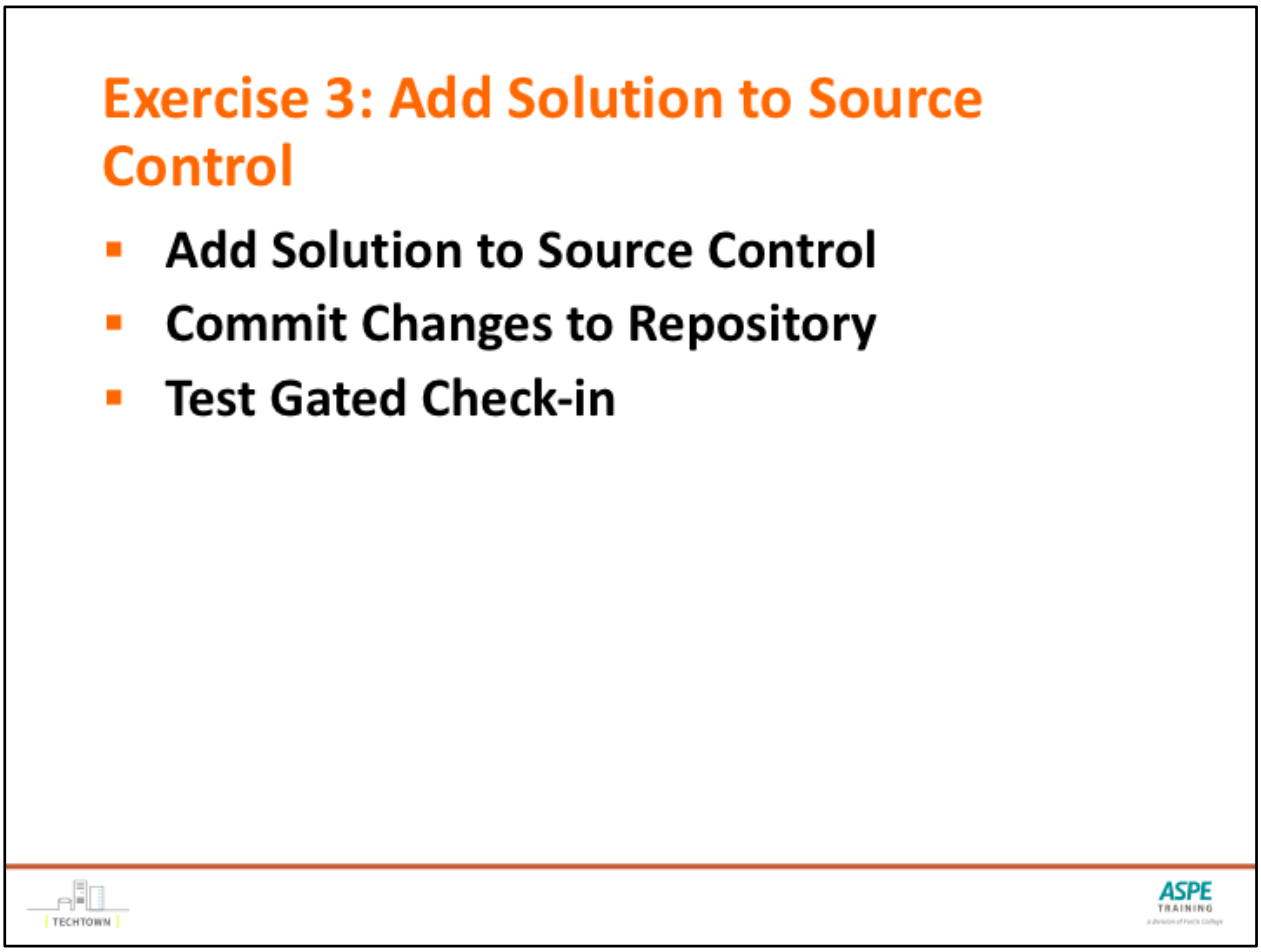
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise 02: Database Deployment Project (Continued)

Build Database Deployment Project

1. Click the **Browse** button, and save your project to the area containing the **Service Reservation WebUI** folder.
2. Accept the rest of the defaults, and click **Start**.
3. When your database project finishes, you will see it in the **Solution Explorer** along with the **WebUI** and the **WebUI tests**.
4. In the database project, expand the folder, open the **dbo** folder, and expand the table folder.
5. You should see a **SQL Server** deployment script for each one of the **SQL Server** tables that was created when we registered the user earlier.
6. Click on the **Team Explorer** Tab, and notice that you already have the **Map and Get.**
7. Accept the defaults, and click the **Map and Get** button.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 3: Add a Solution to Source Control
In this exercise, we will add a solution to source control, test gated check in, and ensure changes to the repository can be seen by the entire team.

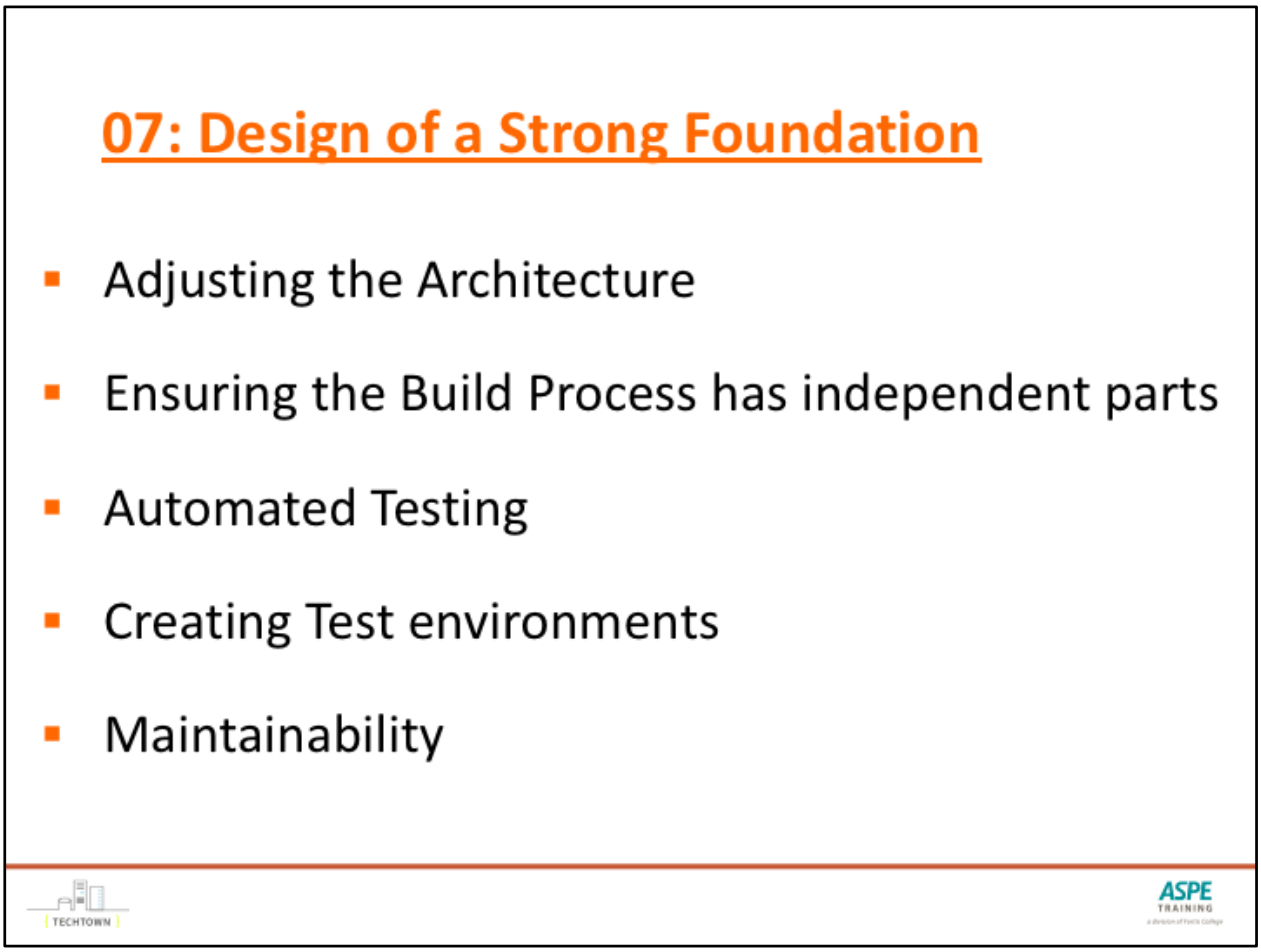
Add Solution to Source Control
In Visual Studio we will commit our changes to Version Control Triggering a Continuous Integration Build

1. Navigate back to the **Solution Explorer**, right click on the **Solution Service Reservations 3** projects, and choose **Add Solution to Source Control**.
2. When the **Add Solution Service Reservations to Source Control** dialogue comes up, locate Service Reservation in the menu, accept the defaults and click OK. *(You should now see plus signs next to the three projects in your solution).*
3. Go back to your browser, and click on the **Work** tab to view the work items in our backlog. If you want, create one work item here.
4. In the Solution Explorer, right click on the **Solution**, and choose check in.
5. **Your check in should get rejected, because we purposefully introduced a defect in it.**
6. Click on the Home icon in Team Explorer, and choose Work Items.
7. After some time, you will see a bug assigned to you…

>>> Due to gated check-in, the code failed to check in
>>> Due to setting we did in pipeline, “assign work item on failure”, after the check-in failed, it assigned a bug to you

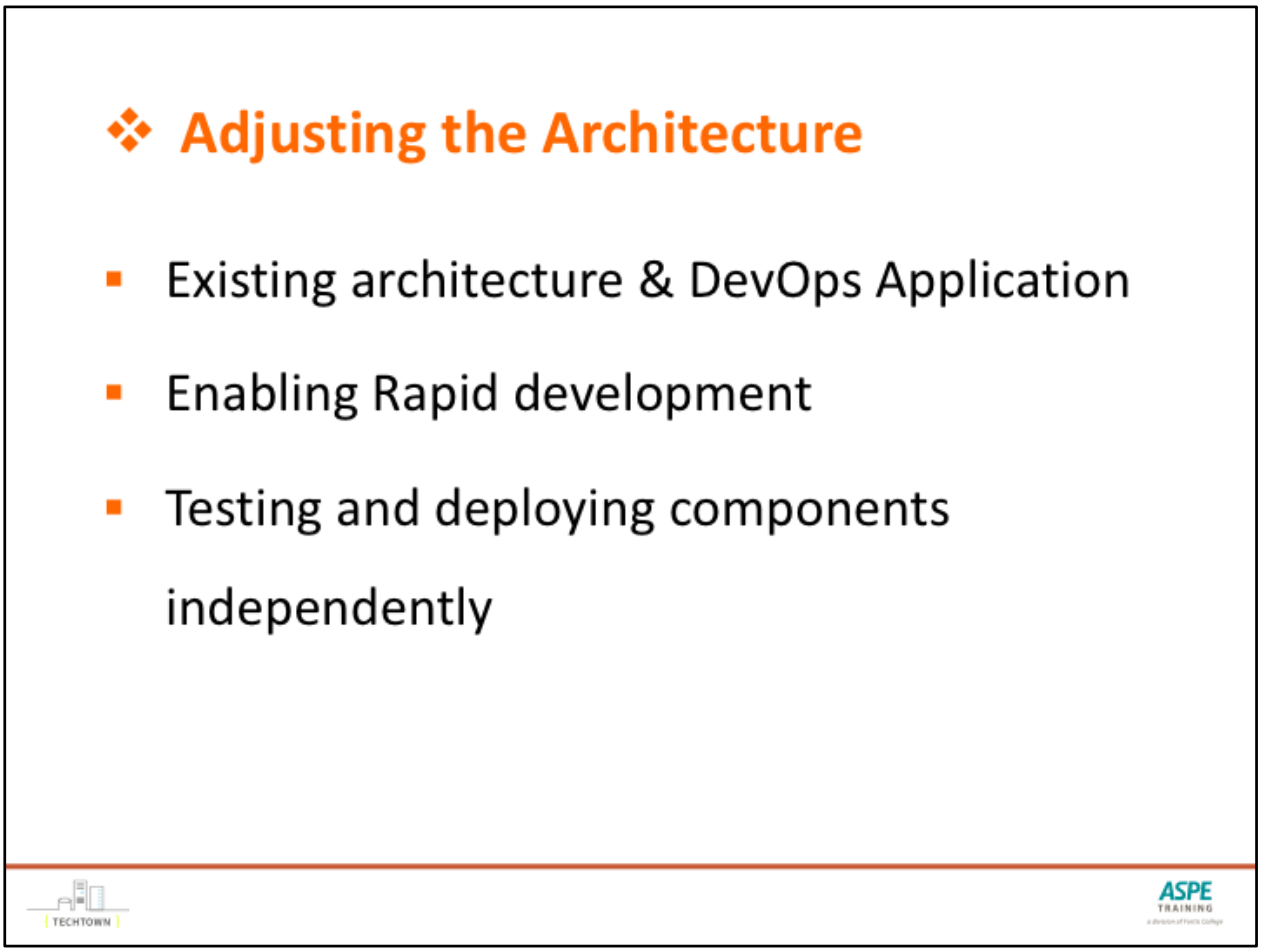
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



DevOps and Agile principles will not be effective without a solid foundation, which requires continuous maintenance through testing, auditing, and constant review of submitted code. This chapter will cover the various processes that can be utilized to identify bugs, uncover errors, and find defects during the build process.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Existing architecture & DevOps Application
Advantages from DevOps and Agile can not be fully obtained without a clean, well defined architecture. Unfortunately, most organizations can not put off improving their architecture, because that would require holding off on process improvements until the architecture was brought up to speed.

Enabling Rapid development
Rapid development will not be possible until different products have been isolated, so improvements can be made without disruption.

Testing and deploying components independently
In most cases, effective … will require minimizing product to product variation so that code can be tested with new features while remaining unchanged.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Existing Architecture & DevOps Application

- Improving the Development Process with DevOps
- Well defined Architecture
  - Containerization
  - Configuration Management
- Smaller and loosely coupled Components

ASPE TRAINING

NOTES:

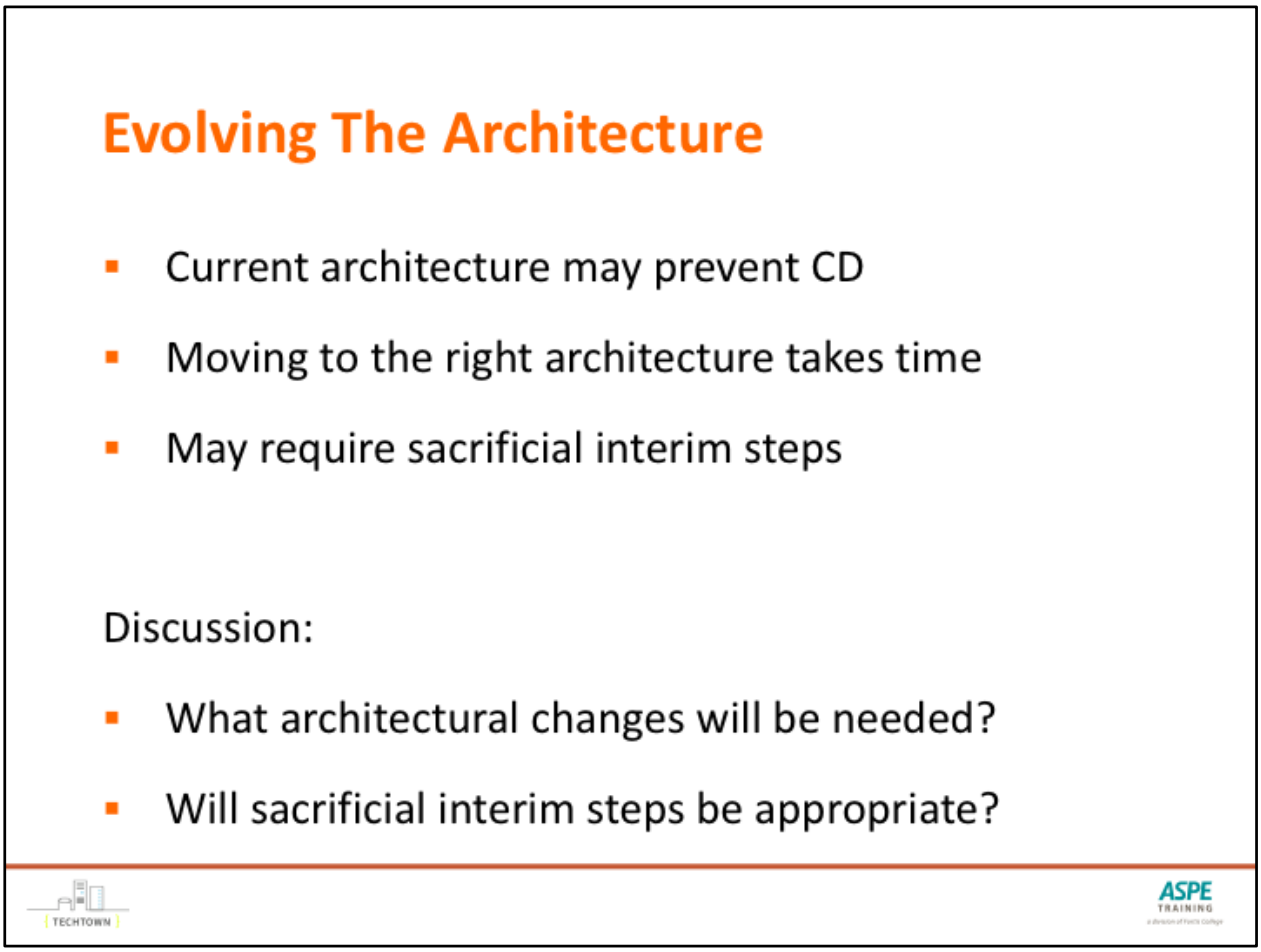
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Enabling Rapid Development

- Creating a solid code Foundation
- Managing Different parts of Architecture
- Overcoming Integration Challenges
- Isolate products by loosely coupled components

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Evolving an architecture can be a long and difficult road. It is common that organizations cannot immediately replace their current inappropriate architecture with the ultimate architecture that fully supports Continuous Delivery (CD). Often multiple intermediate steps are necessary, with each of those steps adding DevOps flexibility and eliminating impediments or bottlenecks. It is not unusual for some of those intermediate steps to be "throw-aways" – things that will not remain in the final architecture, but are necessary to bridge the gaps on your journey there.

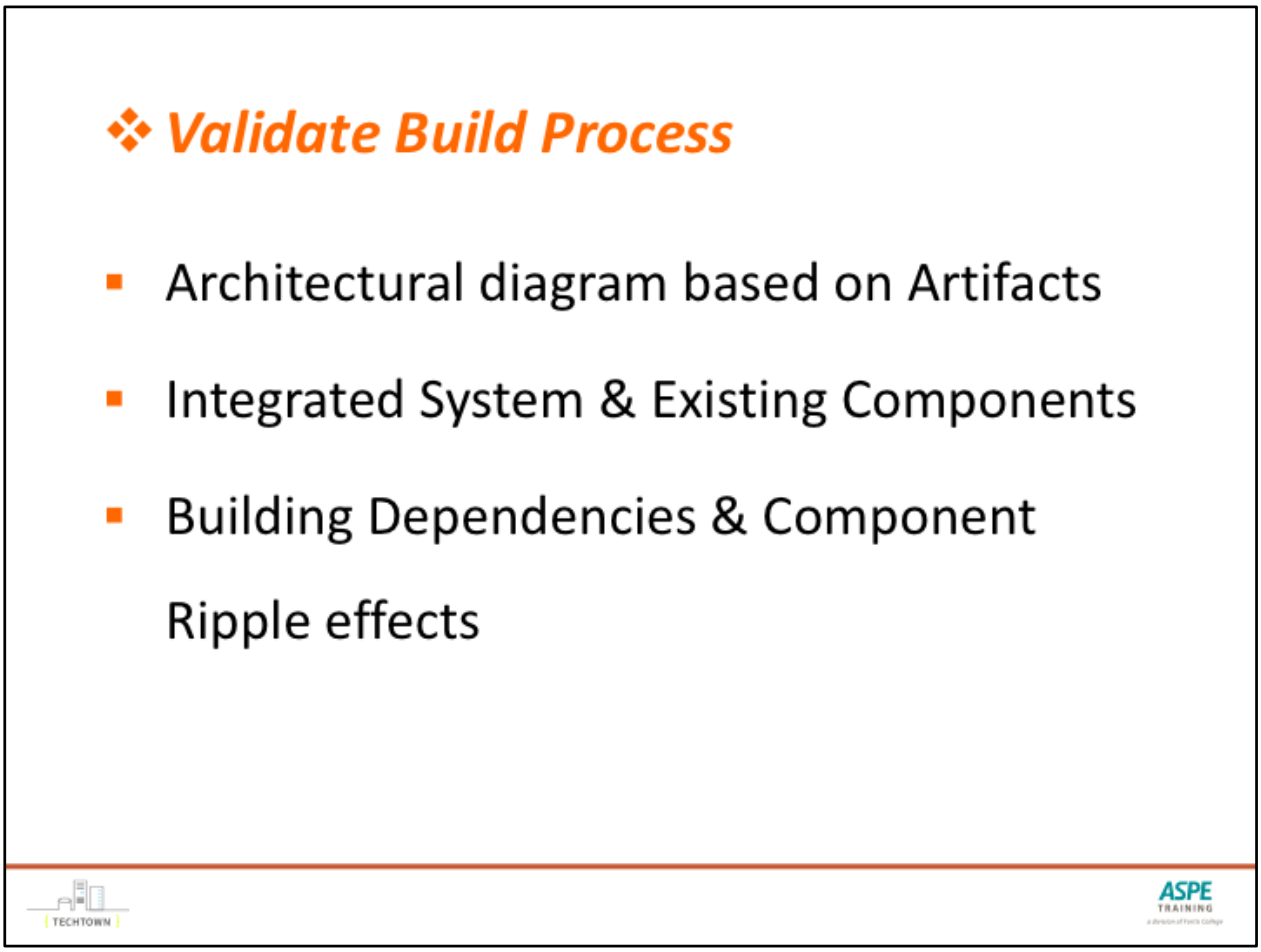
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Testing and Deploying Components *Independently*

- Testing Architecture Components Independently – by loosely coupled components
- Leveraging Code across a range of products
- Keeping the system Stable

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Architectural diagram based on Artifacts
When it comes to artifacts, the architectural challenge will be isolating product variation so code can be leveraged unchanged across the existing product line.

Integrated System & Existing Components
One of the fundamental necessities that large organizations don't have is the evaluation of build process readiness. In order to do so, define and describe the existing components, and use that to make a fully integrated system using these components that ensures the build is green.

Building Dependencies & Component Ripple effects
After automated testing has been set up to ensure green builds, development teams will need to be able to revert back to the previous build when it is not sustainable. If the software does not pass, developers will know they need to fix the build process by modifying existing architecture to ensure that all components can be built and deployed into testing environments independent of others.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Architectural Diagram Based on Artifacts

- Legacy Architecture
- Balancing Architectural changes & Process Improvements
- Structured approaches to Continuous Delivery

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Integrated System & Existing Components

- Minimizing Product Differences
- Evaluating System Stability
- Integrating Existing Components

NOTES:

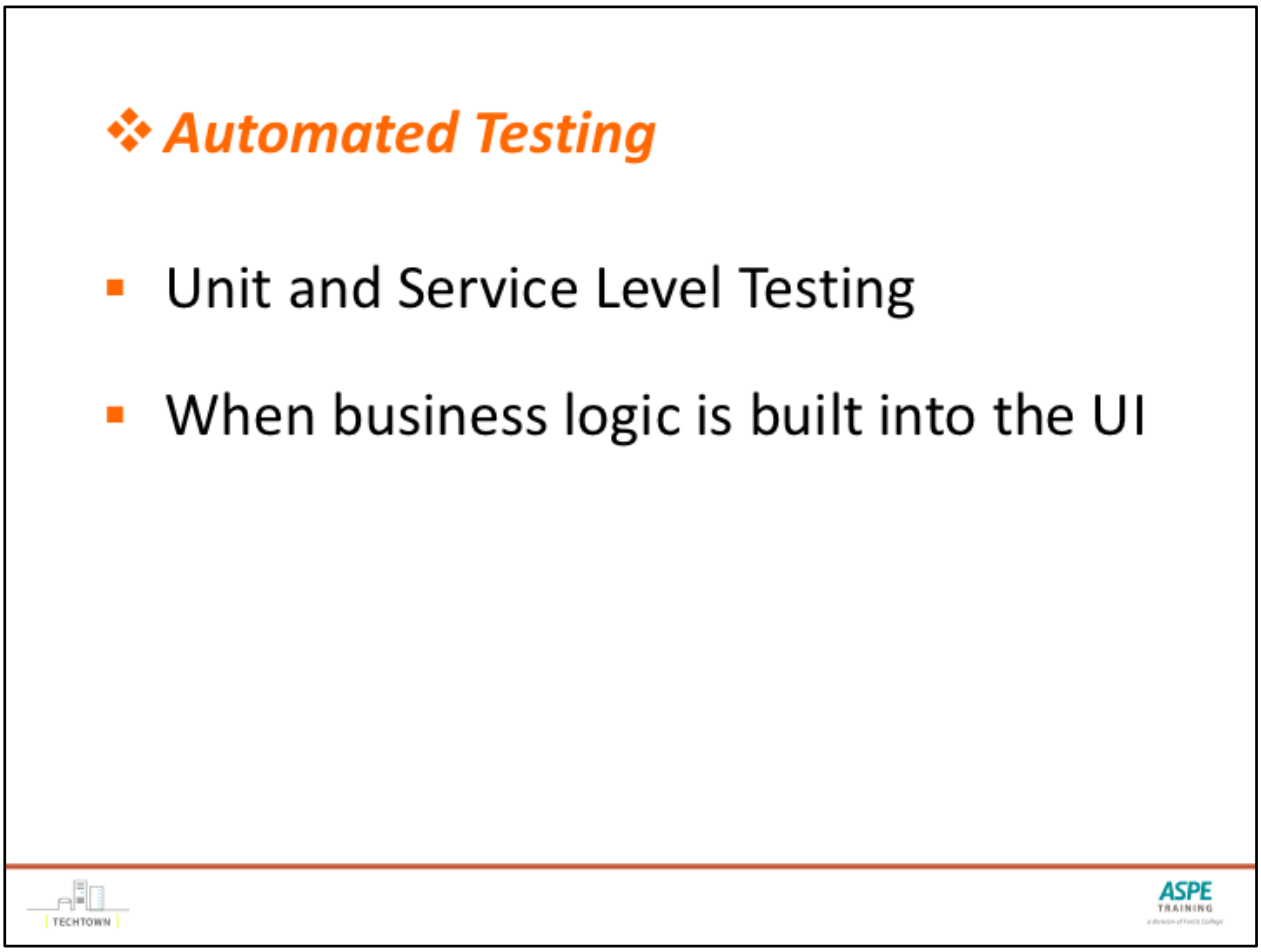
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Building Dependencies & Component Ripple Effects

- Determining what makes a build go red
- Recovering from a bad build
- Getting back to a Green Build

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Unit and Service Level Testing
It is virtually impossible to keep large software systems stable without a very large number of automated tests running at Unit and Service level on a daily basis.

When business logic is built into the UI
When business logic is built into the UI, code stability becomes a major challenge. Tests will still need to be run on a daily basis, but many of them will need to simplified in order to be effective.

System level UI testing
When organizations have tightly couple systems with business logic in the user interface, automated testing can be extremely difficult. It is very important to start with an approach that will make it efficient to deal with thousands of tests on a daily basis.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Unit and Service Level Testing

- Writing good test Automation
- Running Tests on a Daily Basis
- Dealing with large numbers of Tests efficiently

NOTES:

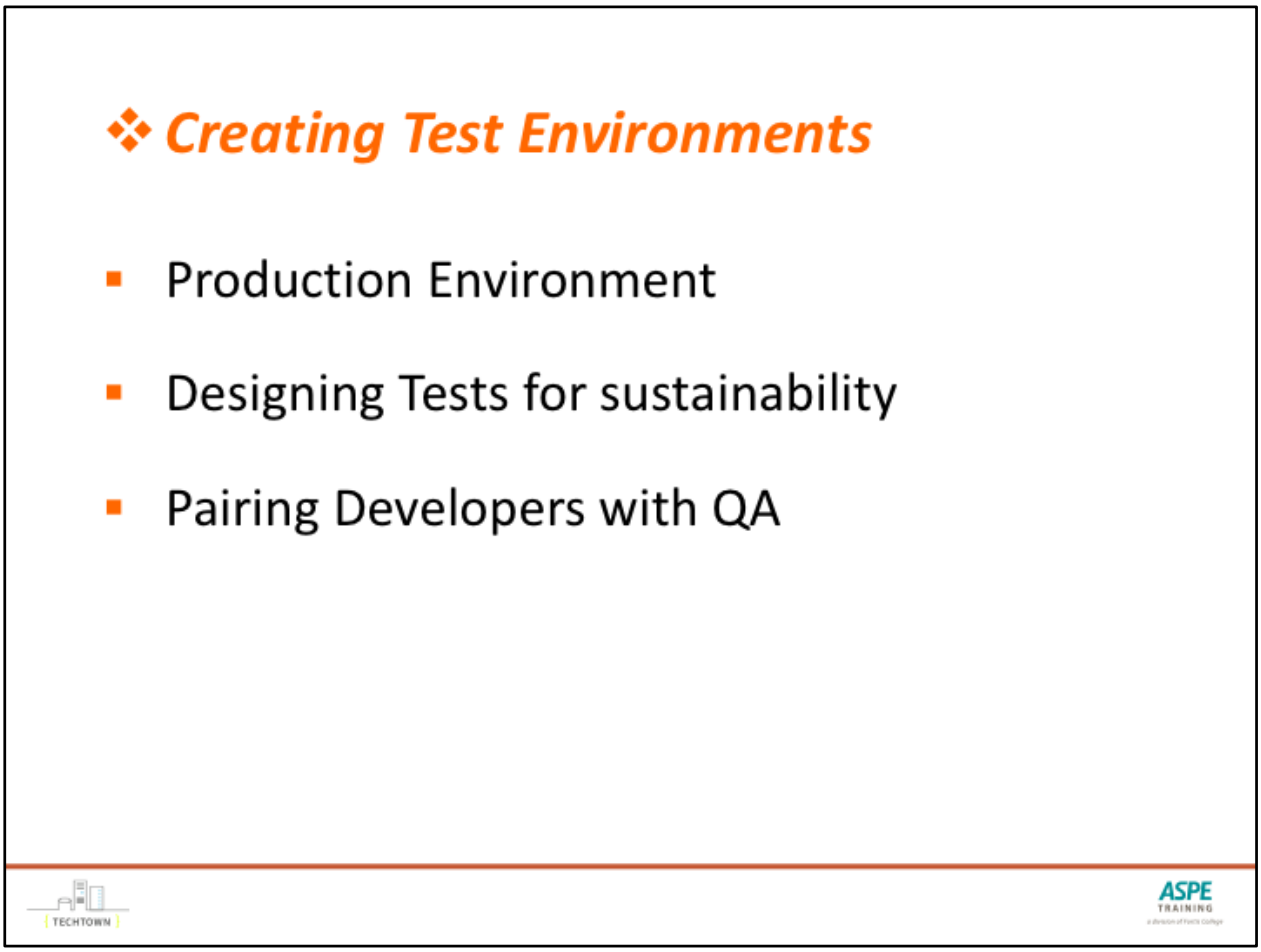
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

When Business Logic is Built into the UI

- The challenge of Code Stability
- Creating and adhering to a good plan
- Focusing on the Team: Developers working with executives

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Production Environment
In order to be effective, test environments must be as much like production environments as possible. It will also need to be economically feasible to run large numbers of these tests on a regular basis.

Designing Tests for sustainability
One of the big challenges for organizations that delegate test automation tasks to QA is that failures need to be identified as code defects, or test defects.

Pairing Developers with QA
The best approach for creating effective automated tests is to pair a qualified developer with a QA engineer that knows how code is manually tested in the current environment.

The most important point here is to pair an architect and developer with the QA team to write a framework for automated testing.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Production Environment

- Running Automated Tests Cost-effectively
- Increasing Test Coverage
- Test Simulators

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Designing Tests for Sustainability

- Code Defect or Test Defect?
- Manual Testing vs Automated Testing
- Software Updates

NOTES:

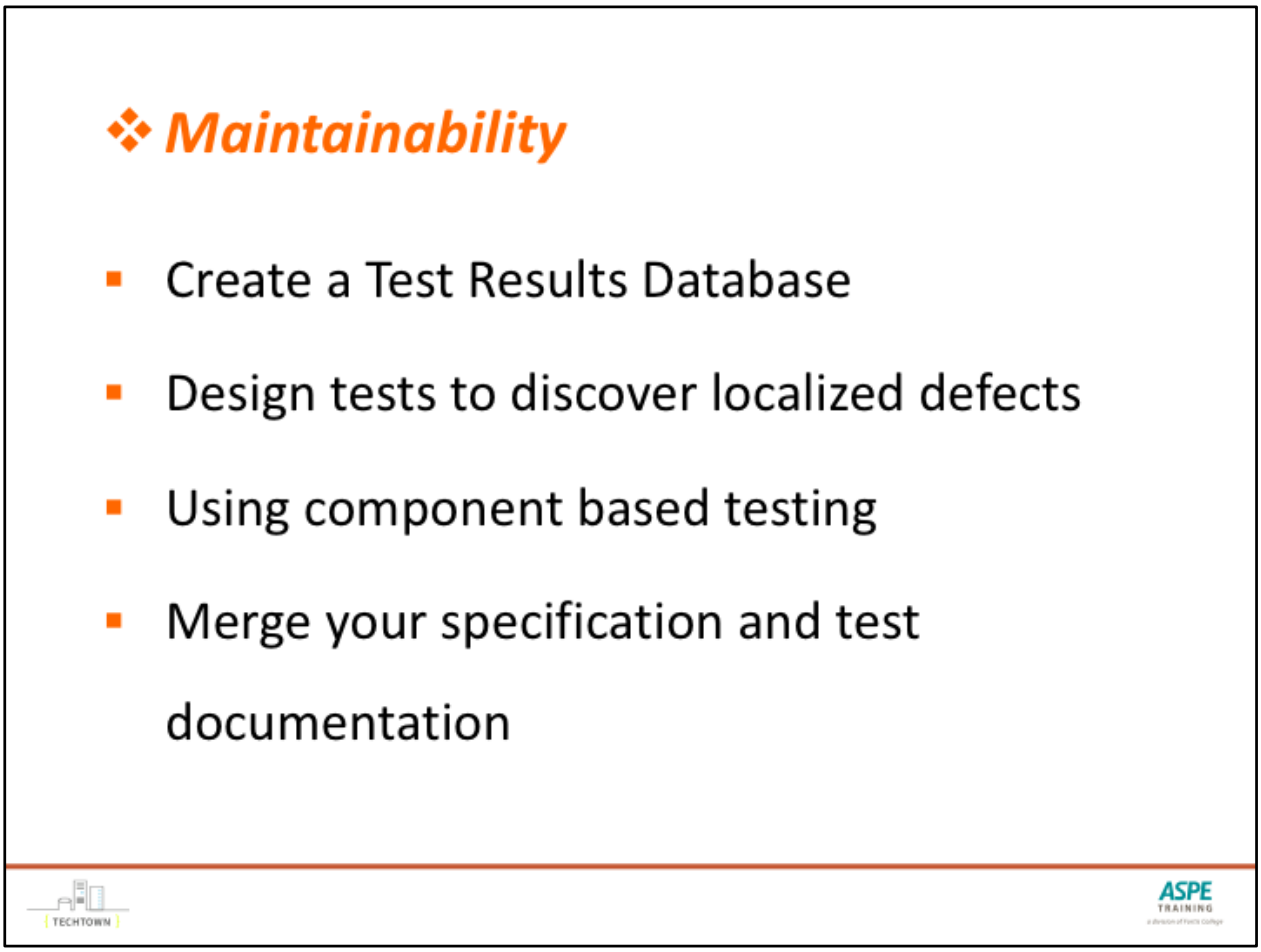
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Pairing Developers With QA

- Pairing Object oriented expertise with Manual Testing
- Creating the Automated Test Framework
- Increase the test coverage

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Create a Test Results Database
Test results databases should retain information that helps move the project forward.
Factors include stable and maintainable framework, test results that are managed and reported, and tracking the number of tests required for statistical analysis, and grouping tests to localize code issues.

Design tests to discover localized defects
A common mistake among large organizations when switching from manual to automated testing is to take their existing QA teams and simply have them automate existing manual scripts. It is now possible to develop automated testing that isolates various system components while it is fully integrated and deployed.

Using component based testing
Positive changes occur when the framework allows for the isolation of existing component functionality.

Merge your specification and test documentation
Tests need to be written to quickly localize which team the offending code came from. Sometimes tests fail because the test design is broken, and documentation can be used to decrease the likelihood that it is a test failure and not the developers code.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Create a Test Results Database

- Managing Test Results
- Reporting Test Results
- Statistical Approach to Test Pass Rates

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Design Tests to Discover Localized Defects

- Automating existing Manual Scripts
- Localizing the cause of failure
- Requirements for running thousands of tests

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Component Based Testing

- Isolating Components of the System
- Testing designed for specific functionality
- Statistically Analyzing Test Metrics

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Merge Your Specification and Test Documentation

- Associating requirements with Code
- Creating Specifications in an executable form
- Testing to meet Audit Requirements

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise: Identify Deployment Improvements

- Update your Value Stream Map to reflect Automating Deployment
  - Estimate new Lead Times and %CAs
- Identify how Automating Deployment improves your Deployment process

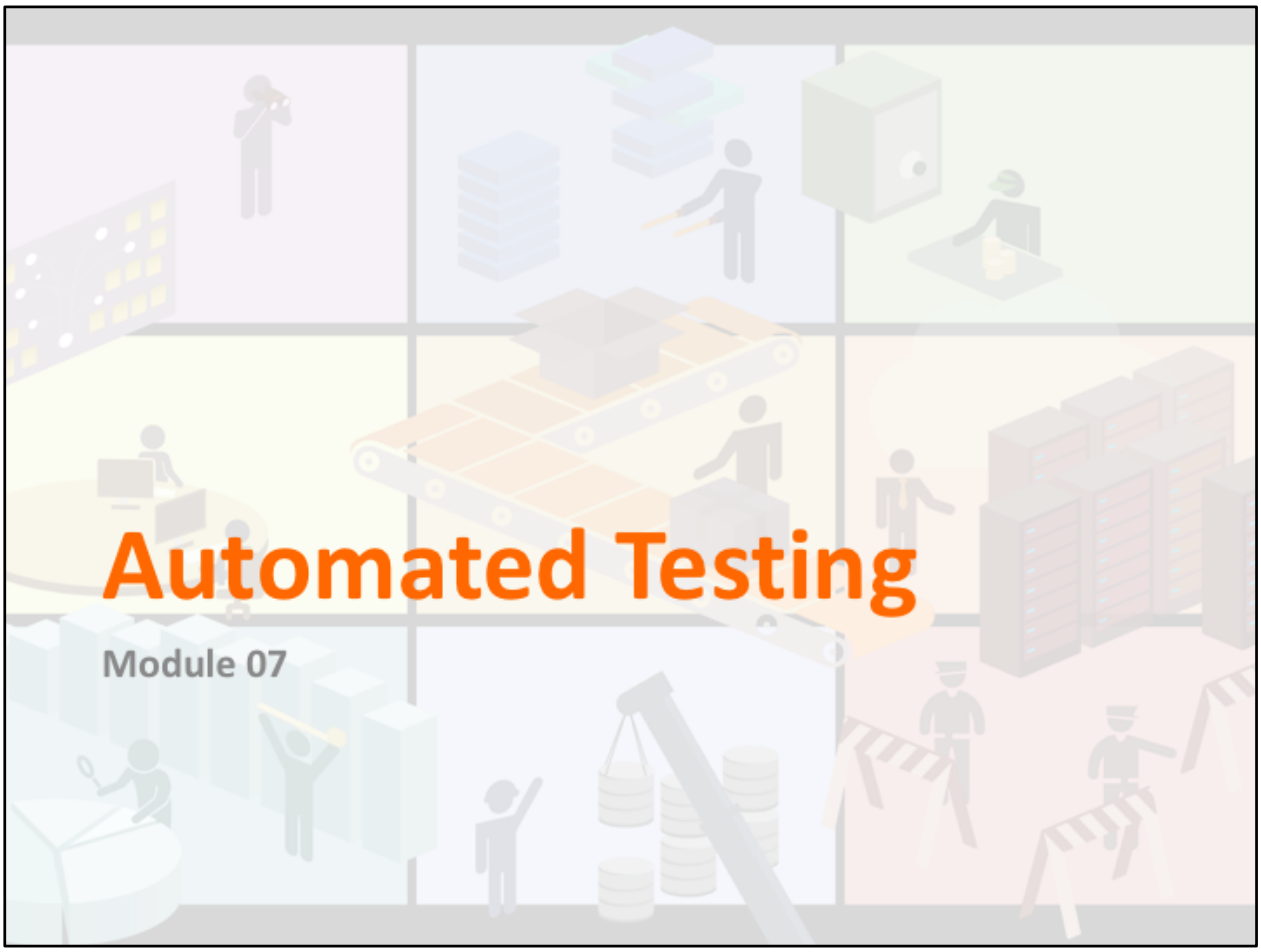
Consider architectural changes also

TECHTOWN

ASPE TRAINING

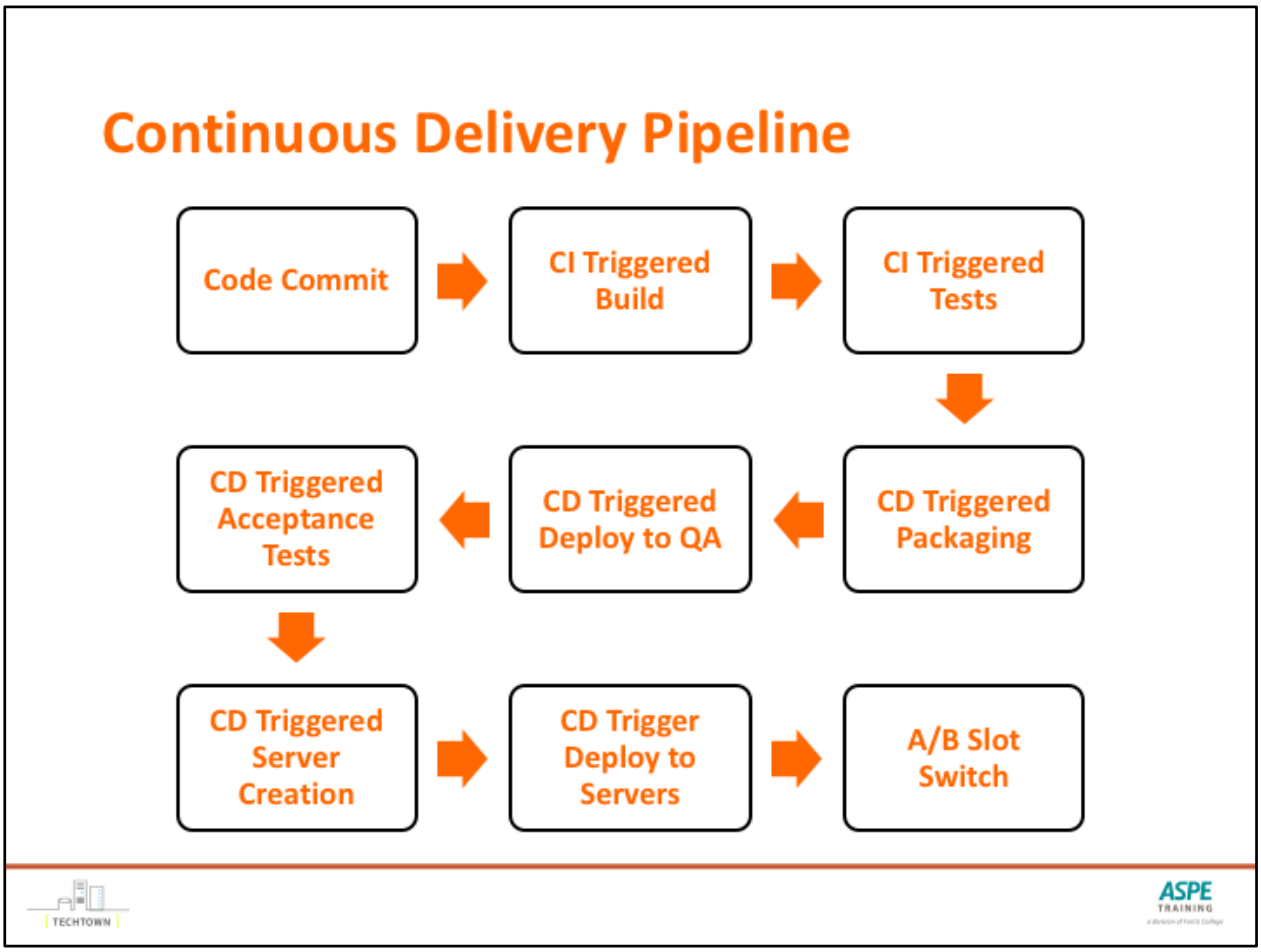
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

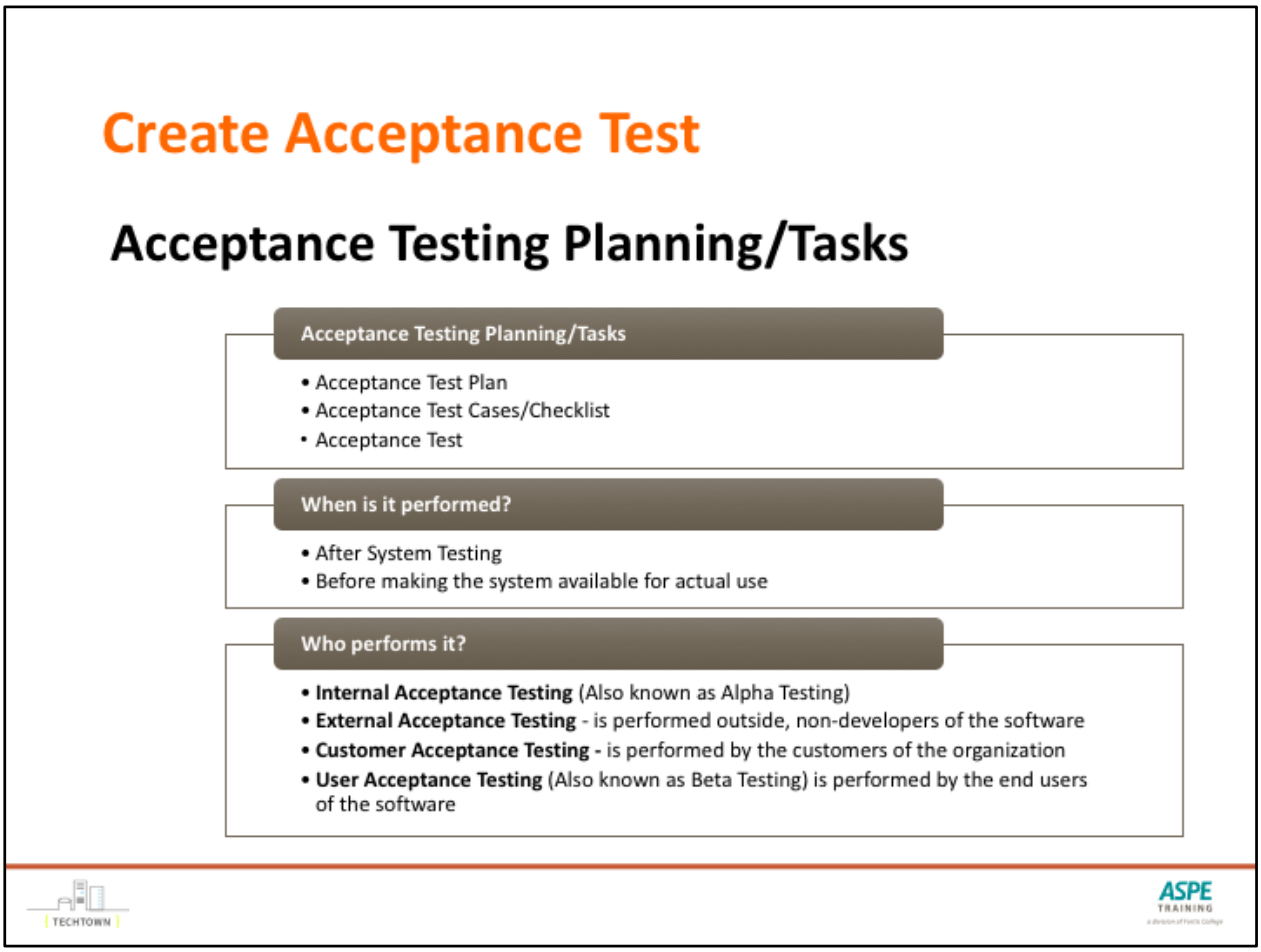
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Module 7: Automated Testing

- Creating Acceptance Tests
- Creating Unit Tests
- Automating Unit Tests
- Automating Capacity Testing
- Parallel Testing

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



METHOD

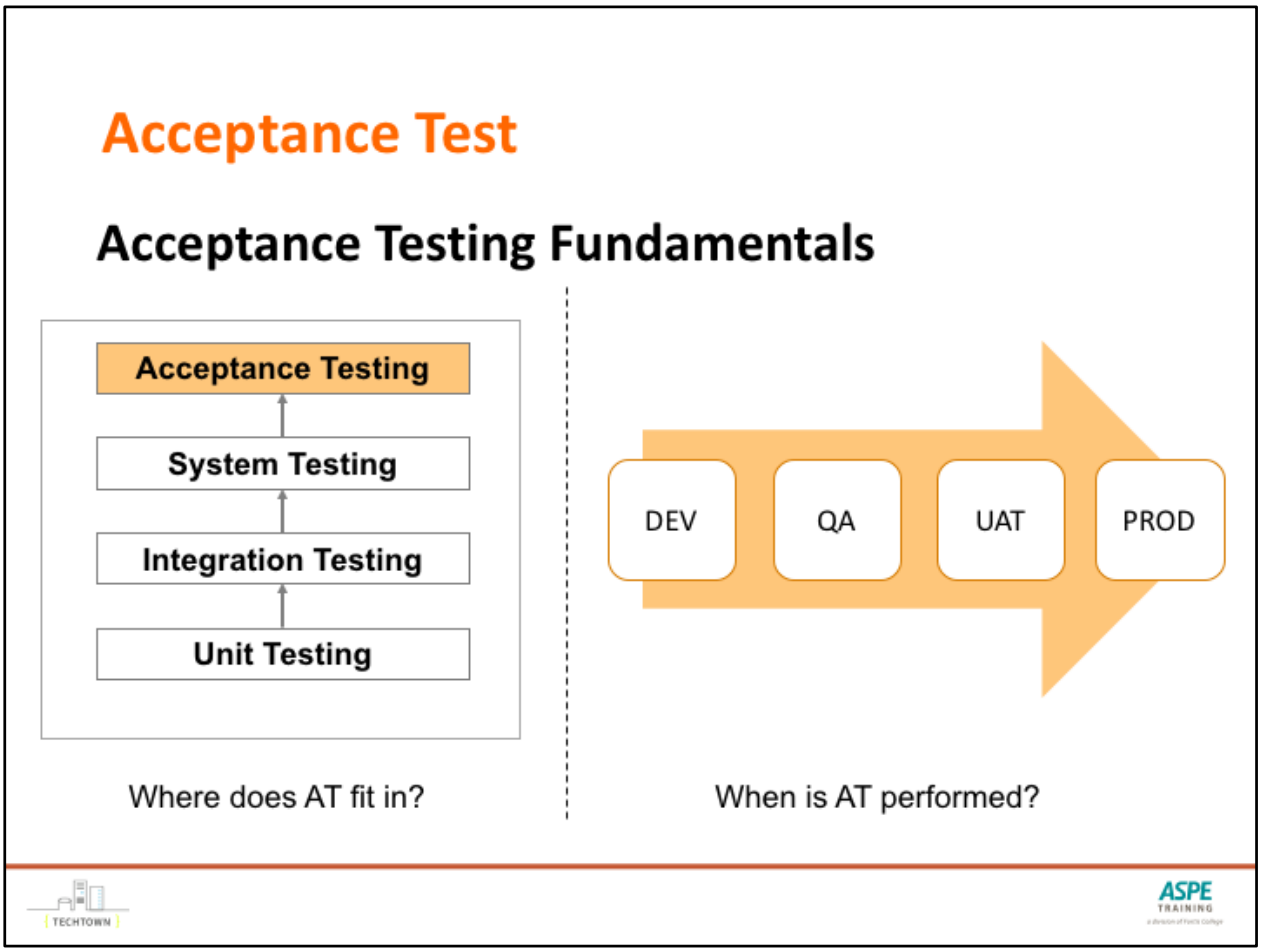
Usually, Black Box Testing method is used in Acceptance Testing. Testing does not normally follow a strict procedure and is not scripted but is rather ad-hoc.

TASKS

- Acceptance Test Plan
    - Prepare
    - Review
    - Rework
    - Baseline
- Acceptance Test Cases/Checklist
    - Prepare
    - Review
    - Rework
    - Baseline
- Acceptance Test
    - Perform

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



METHOD

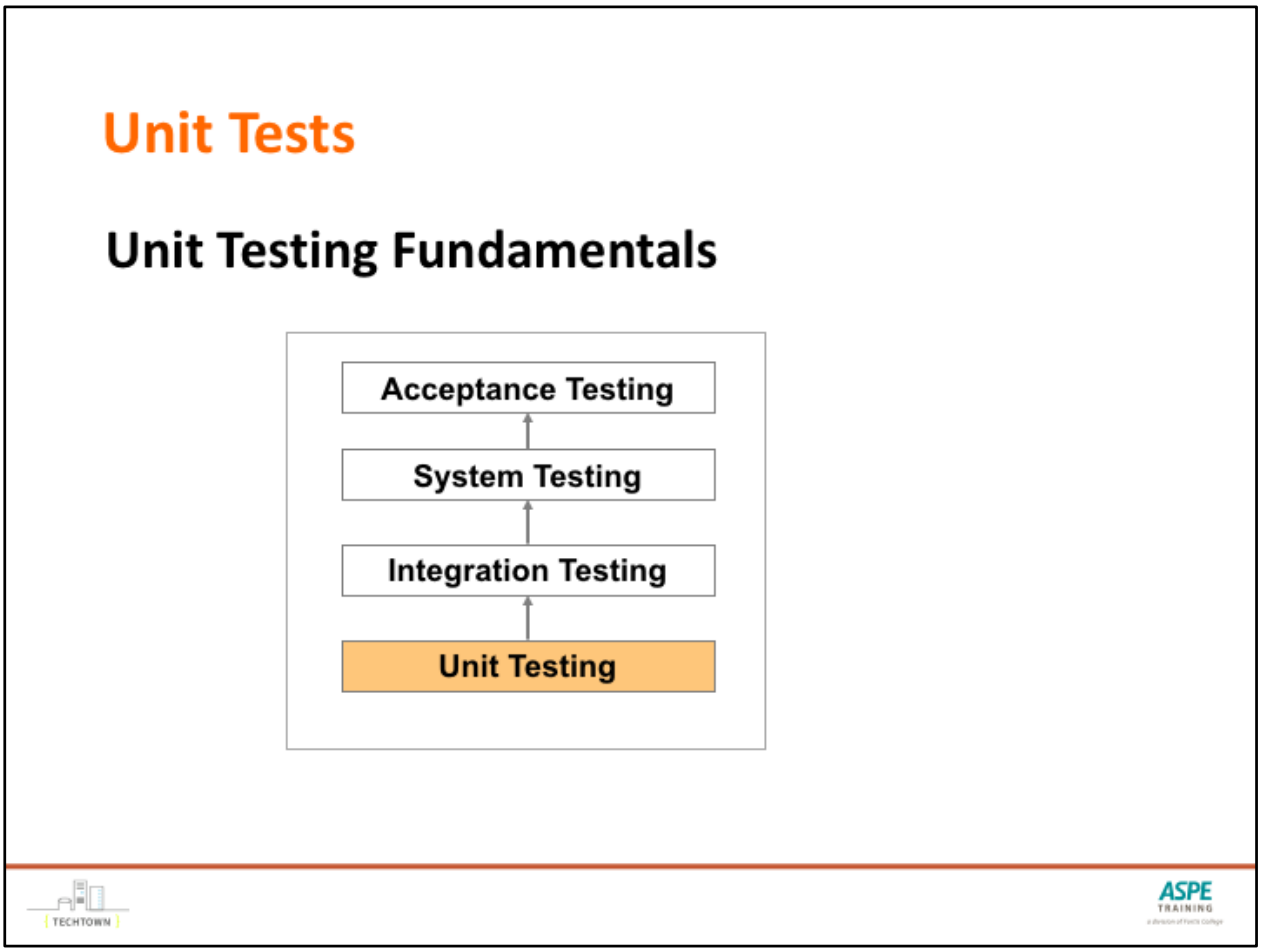
Usually, Black Box Testing method is used in Acceptance Testing. Testing does not normally follow a strict procedure and is not scripted but is rather ad-hoc.

TASKS

- Acceptance Test Plan
  - Prepare
  - Review
  - Rework
  - Baseline
- Acceptance Test Cases/Checklist
  - Prepare
  - Review
  - Rework
  - Baseline
- Acceptance Test
  - Perform

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



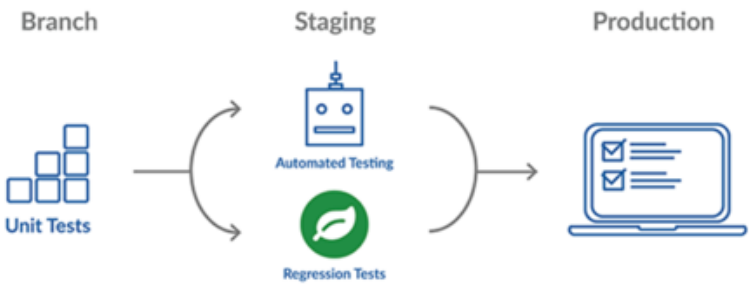
Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed.

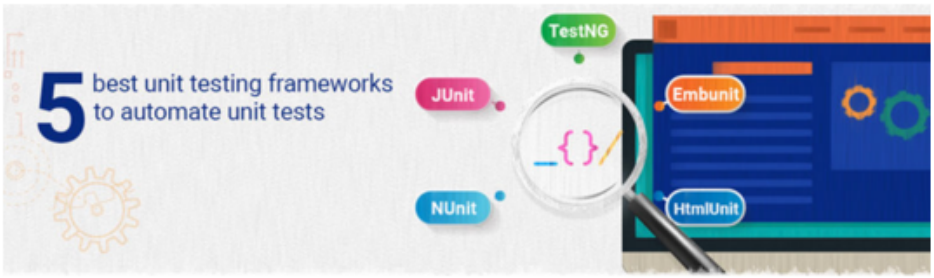
A unit is the smallest testable part of software. It usually has one or a few inputs and usually a single output. In procedural programming a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.)

Unit testing frameworks, drivers, stubs, and mock/fake objects are used to assist in unit testing.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Automating Unit Tests





Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Capacity Testing

- The primary goal of testing is to establish the benchmark behavior
- The two types of capacity tests are
  - Performance and
  - Load/stress tests

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Capacity Testing

The figure shows flow of execution in capacity testing approach.

TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

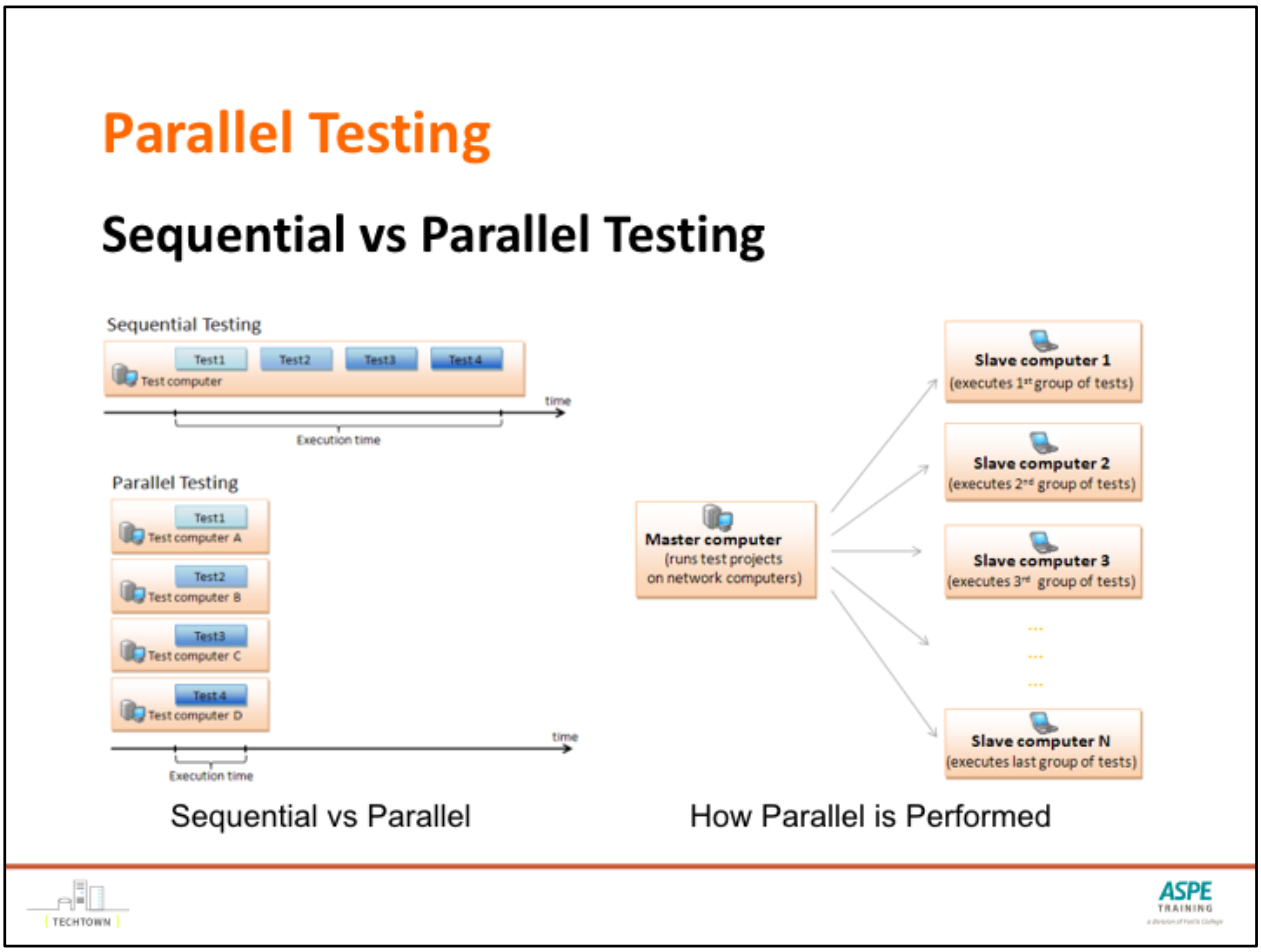
Automated Capacity Testing

Top 15 Performance Testing Tools

- WebLOAD
- LoadUI NG Pro
- Apica LoadTest
- LoadView
- Apache JMeter
- LoadRunner
- Appvance
- NeoLoad
- LoadComplete
- WAPT
- Loadster
- LoadImpact
- Rational Performance Tester
- Testing Anywhere
- OpenSTA
- QEngine (ManageEngine)
- Loadstorm
- CloudTest
- Httperf

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

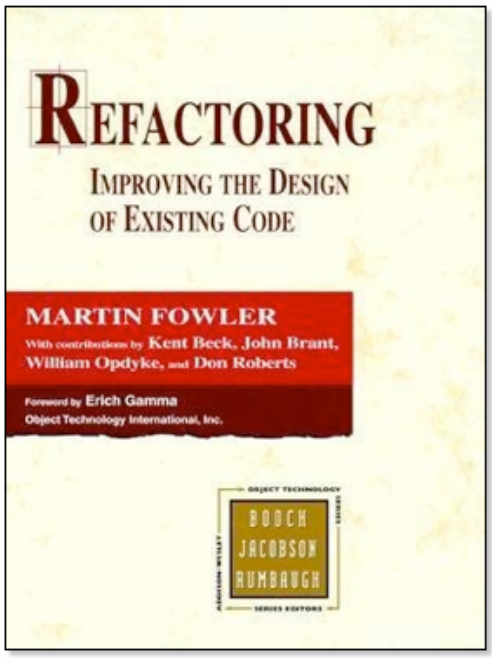


Parallel Testing

Parallel testing means testing multiple applications or subcomponents of one application concurrently to reduce the test time. Parallel tests consist of two or more parts (projects or project suites) that check different parts or functional characteristics of an application. These parts are executed on individual computers simultaneously. The ability to test more than one application part "in parallel" allows you to significantly reduce the test time and test your applications more efficiently.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Refactoring



Refactoring is an ongoing activity of improving the internal structure of code without making externally visible changes to the functionality

TECHTOWN

ASPE TRAINING

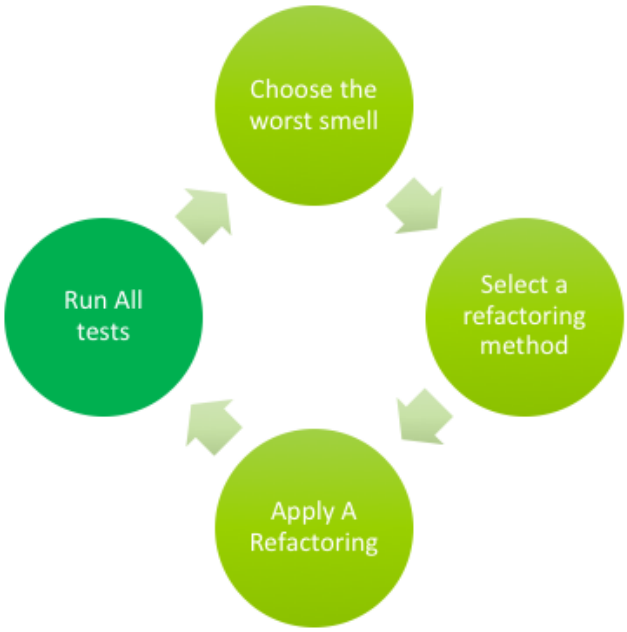
- Refactoring is a long-term, cost-efficient, and responsible approach to software ownership.
- Code clarity degrades over time the maintenance process loses design intent.

Refactoring Rules of The Road

- Tests must exist.
- Add them if they are missing
- All tests should be green
- Code should be left in better condition than you found it.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Refactoring Cycle



Pre-conditions

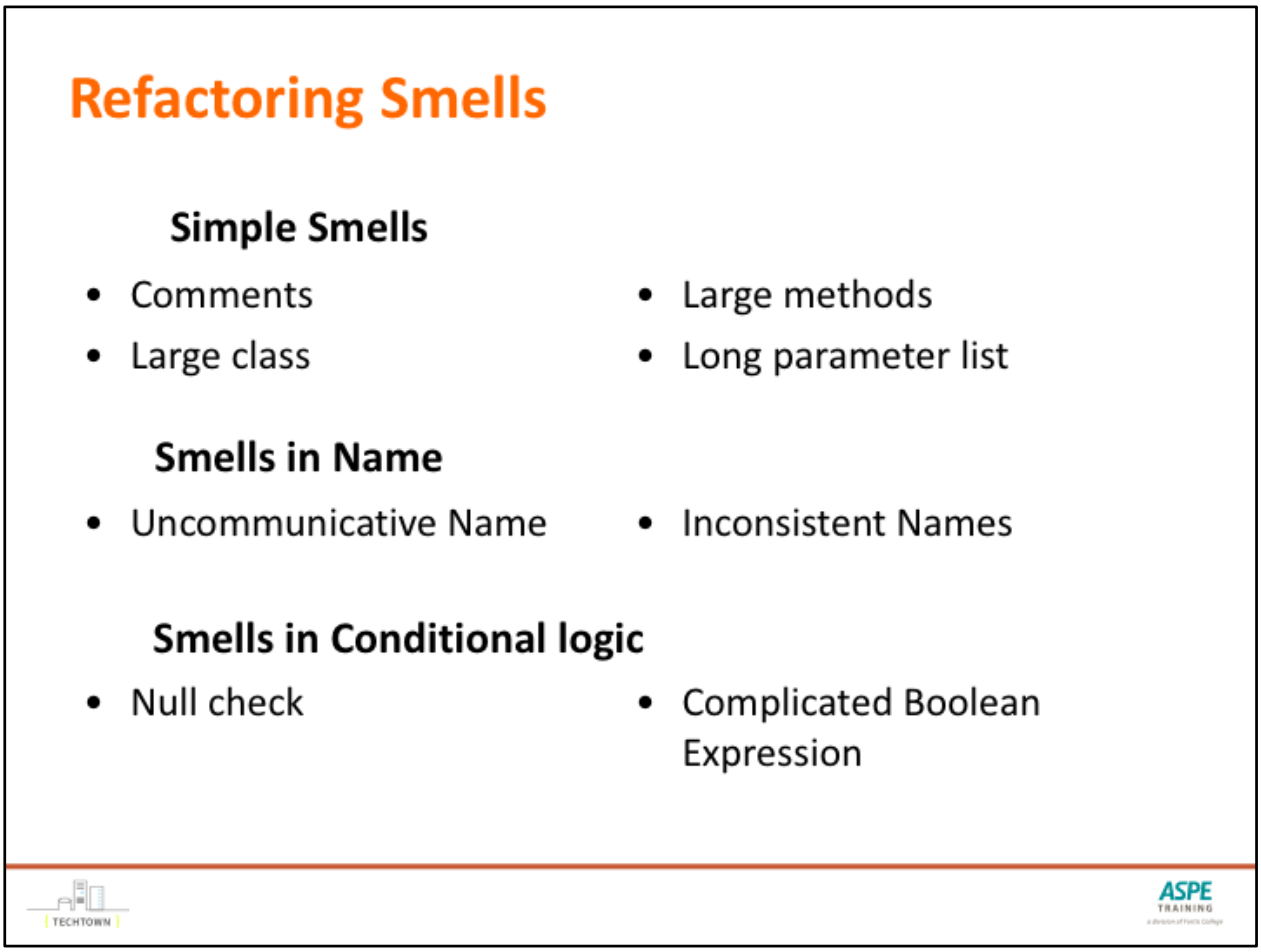
- All Tests are Up-to-date, If tests need to be updated do so prior to refactoring
- All Tests are **Green**

Post-conditions

- Refactored code should be as good as the pre-existing code or better
- All Tests are **Green**

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Comments

Comments are usually created with the best of intentions, when the author realizes that his or her code is not intuitive or obvious. If you feel that a code fragment cannot be understood without comments, try to change the code structure in a way that makes comments unnecessary.

Treatment

Consider using Extract variable or Extract method

Long Method

A method contains too many lines of code. Generally, any method longer than ten lines should make you start asking questions.

Treatment

if you feel the need to comment on something inside a method, you should take this code and put it in a new method.

Large Classes

Classes usually start small. But over time, they get bloated as the program grows.
As is the case with long methods as well, programmers usually find it mentally less taxing to place a new feature in an existing class than to create a new class for the feature.

Treatment

When a class is wearing too many (functional) hats, think about splitting it up:

Long Parameter List

Long list of parameters might happen after several types of algorithms are merged in a single method. A long list may have been created to control which algorithm will be run and how.

Treatment

Check what values are passed to parameters. If some of the arguments are just results of method calls of another object, use Replace Parameter with Method Call. This object can be placed in the field of its own class or passed as a method parameter.

NOTES:

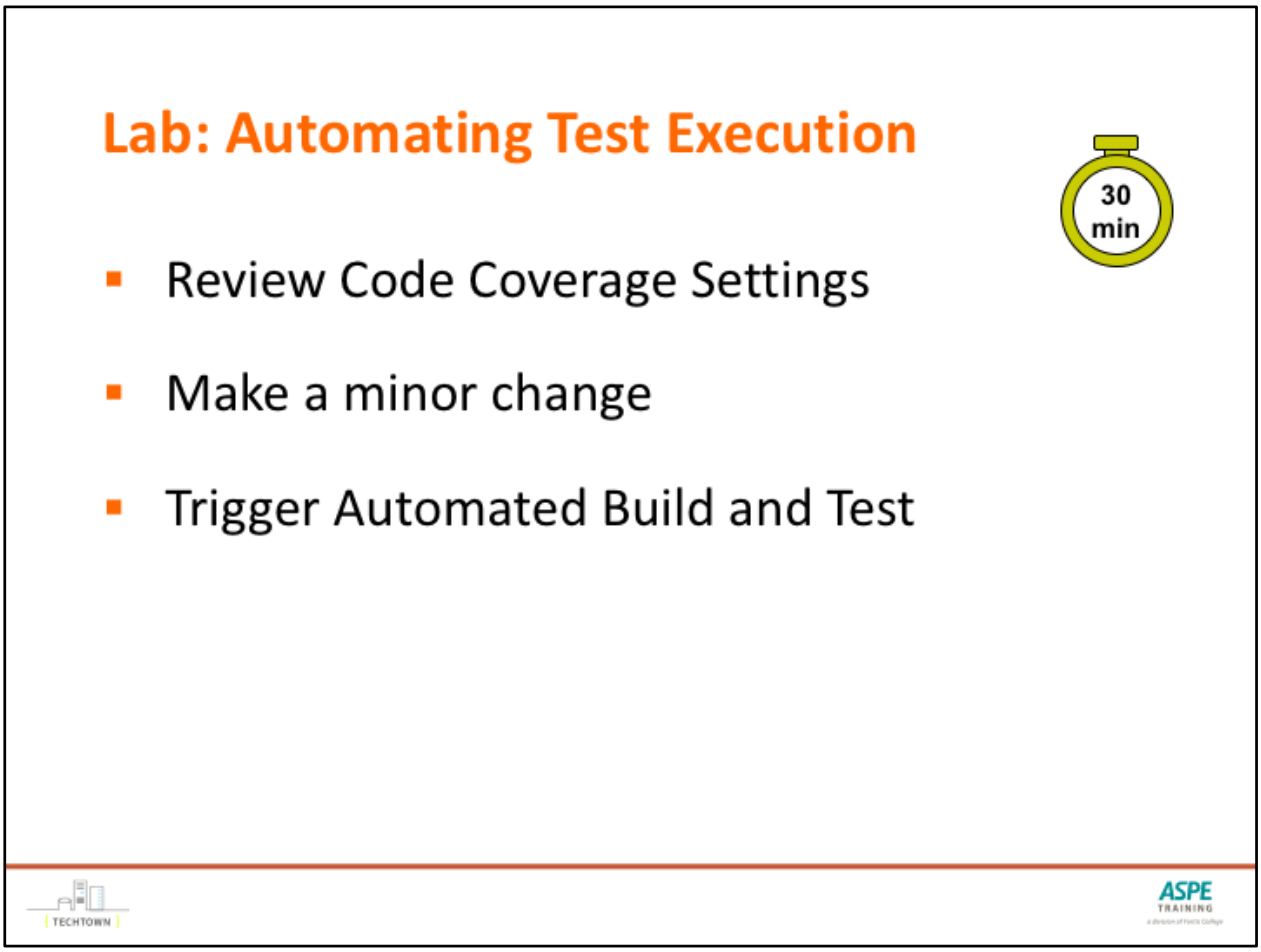
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

When Are we Done?

- Passes all the tests.
- Has no duplicated logic.
- States every intention important to the programmers.
- Has the fewest possible classes and methods.

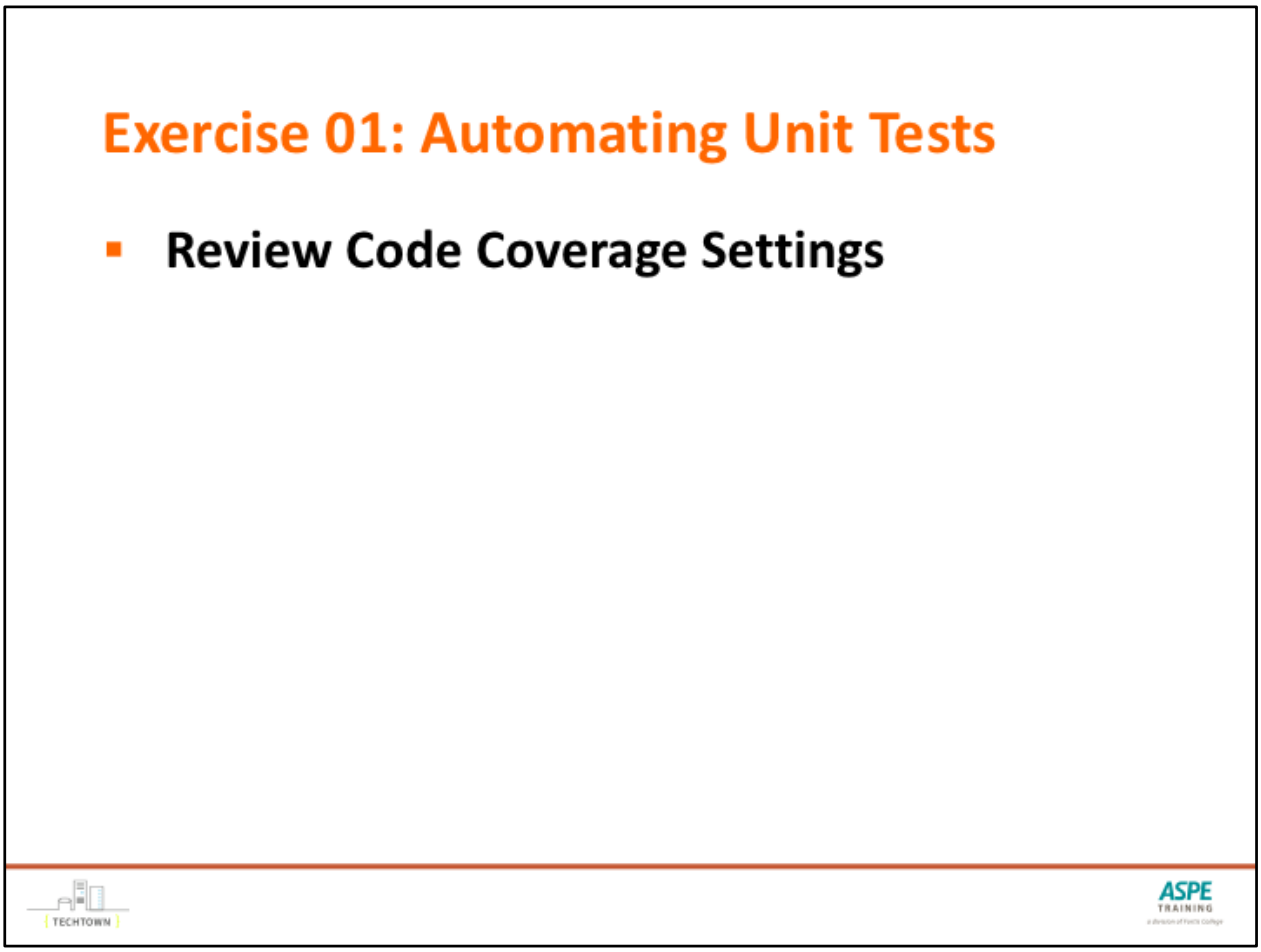
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In this lab we will review code coverage results after making a minor change and triggering an automated ***Continuous Integration*** build and test.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



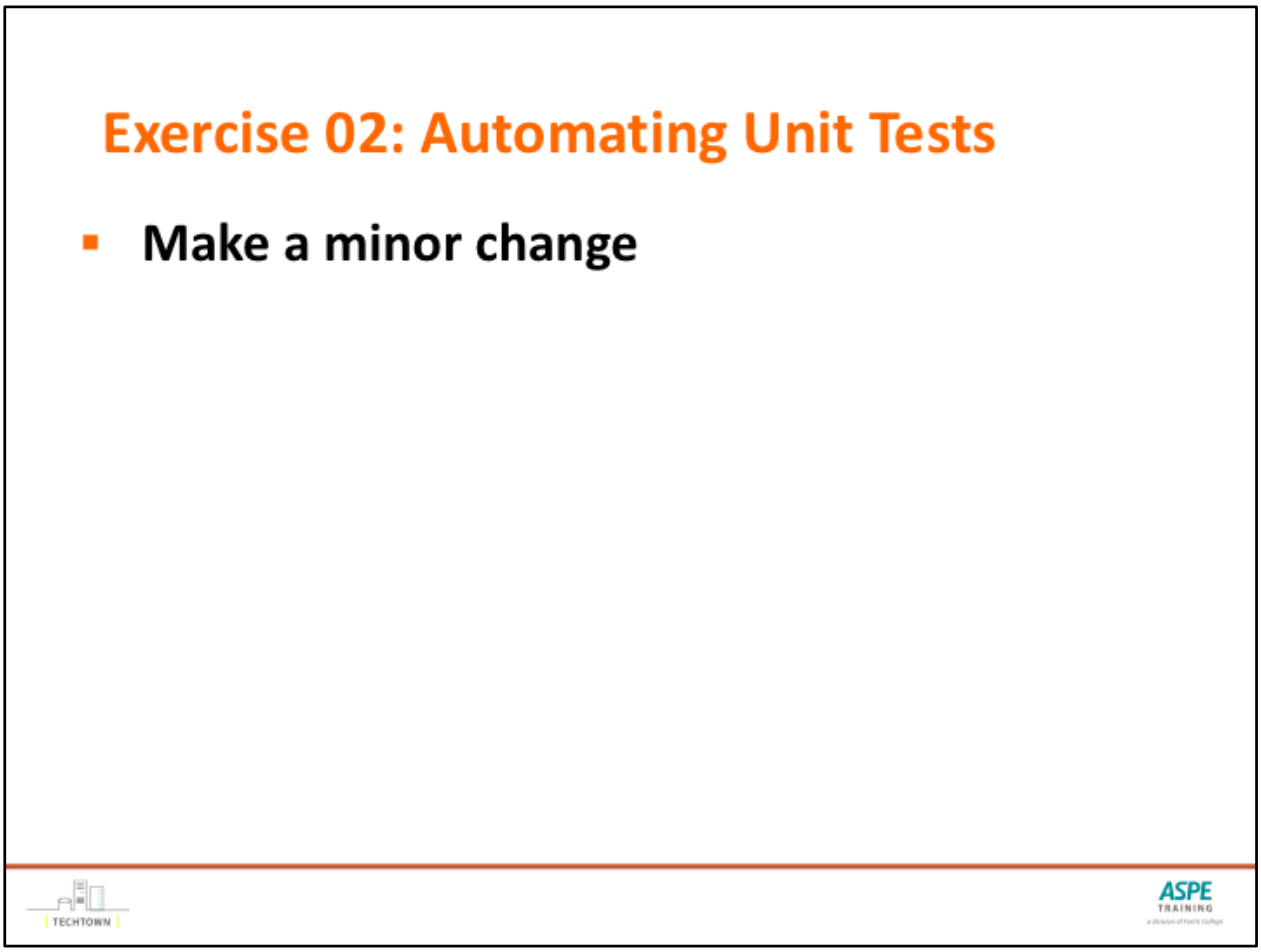
Exercise 1: Automating Unit Tests

In this exercise, we will enable code coverage, so we can estimate the value of our current testing criteria.

Review Code Coverage Settings

1. Navigate to the **Team Foundation Services** dashboard.
2. Click on the **Build and Release** tab, and select **Build.**
3. In the **Service Reservation ASP.NET-CI** project, Click on the ellipses link under the Status menu option and choose **Edit**
4. In the Process menu, choose **Test assemblies**
5. Scroll down, and check the **Code coverage** enabled checkbox.
6. Check to make sure that the **Run only impacted tests** checkbox is NOT checked.
7. Expand the **Save & Queue** menu, and choose **Save** *(do not save and queue).*
8. Accept the defaults, and click the **Save** button

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2: Automating Unit tests

In this exercise, we will make some changes to our code, and automate code coverage.

Make a minor change

1. In **Microsoft Visual Studio,** navigate back to the **Service Reservations WebUI** project.
2. In the **Solution Explorer**, click on the **HomeController cs.**
3. In the corresponding code window, scroll down until you see:

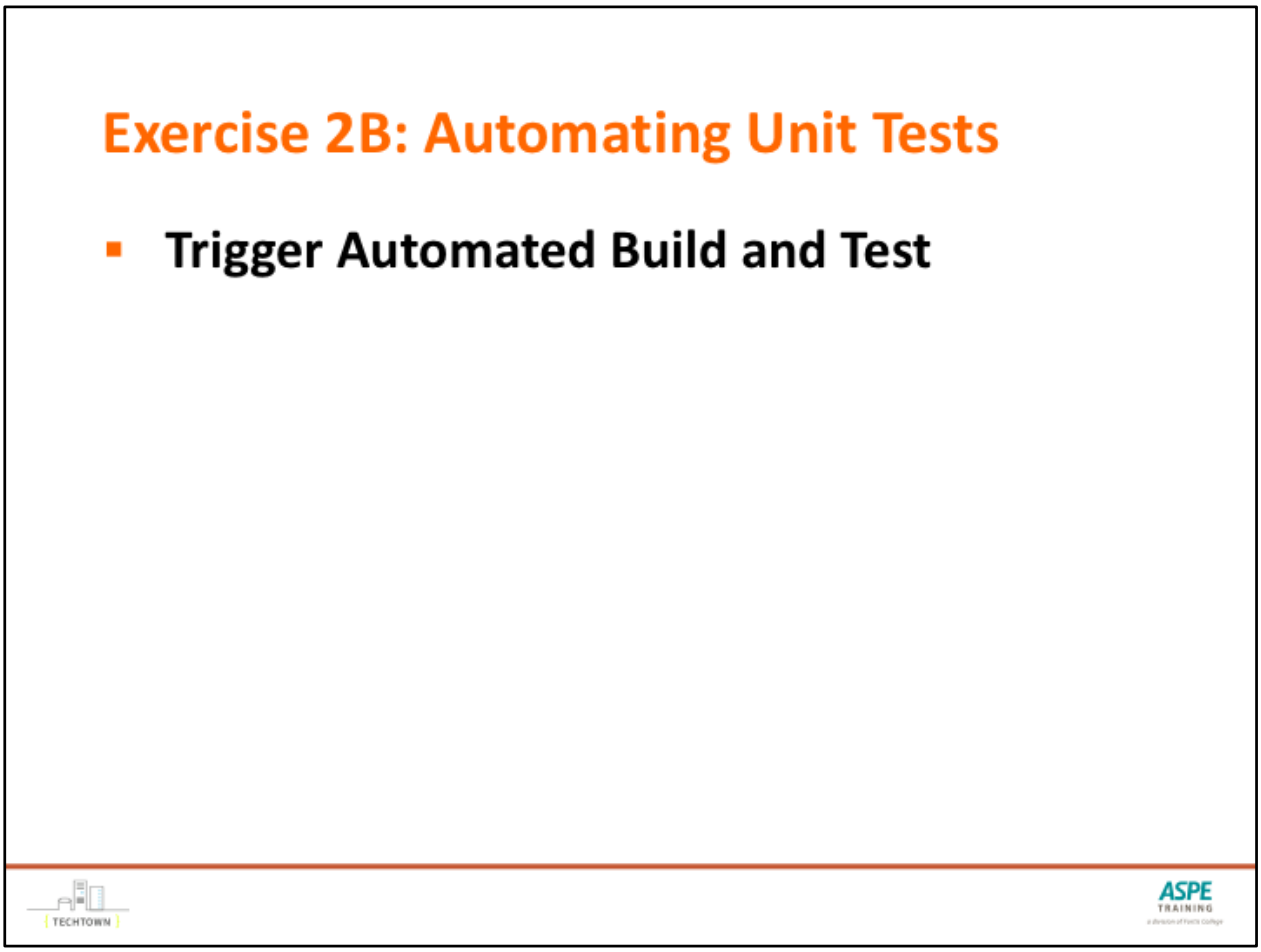
```
public ActionResult About()
{
ViewBag.Message = “Your application description”
return View();
}
public ActionResult Contact()
{
ViewBag.Message = “Your contact page”;
```

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

4. Review the ViewBag.Message property value. (This is where the heading of the page is set)
5. Navigate back to the Solution Explorer, and scroll down to **Service Reservation WebUI.Tests.**
6. Expand the **Controllers** menu, and click on **HomeControllerTest.cs**
7. In the code window, scroll down to:
   // Assert
   Assert.IsNotNull(result);
8. Replace **Assert.IsNotNull(result);** with **Assert.AreEqual("Service Reservation Contact page.", result.ViewBag.Message);**
   *Note: It may be easiest to copy **Assert.AreEqual("Your Application Description page.", result.ViewBag.Message);** and change the text to Service Reservation Contact Page instead of retyping the whole line*
9. Scroll back up to **"Your Application description page."**, and change it to:
   "Service Reservations description page"
   (We are modifying the code to cause the tests to fail)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2B: Automating Unit Tests

In this exercise, we will continue the automation of our Unit tests by creating a trigger that will fire off when changes are applied to our source control repository.

Trigger Automated Build

1. Navigate to the Team Foundation Services Dashboard, and chose the **Build and Release** tab
2. In the **Solution Explorer**, right click on "**Service Reservations WebUI**" **(2 projects),** select the **Check In** menu item, and click the **Yes** button when the dialogue pops up.
3. When the **Gated Check** in dialogue pops up, accept the defaults and click the **Build Changes** button.
4. In the **Solution Explorer**, click on the link "**Your check-in has been queued for validation. Click here to see the status of the validation build."**
5. Click on the **Build Summary** link to view the results.
6. Notice our Tests have failed and our **Gated check-in** has been rejected.
7. Review the Test Results in the Build summary
8. Back in **Visual Studio** in the **Solution Explorer**, click on the **HomeController cs.**

(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

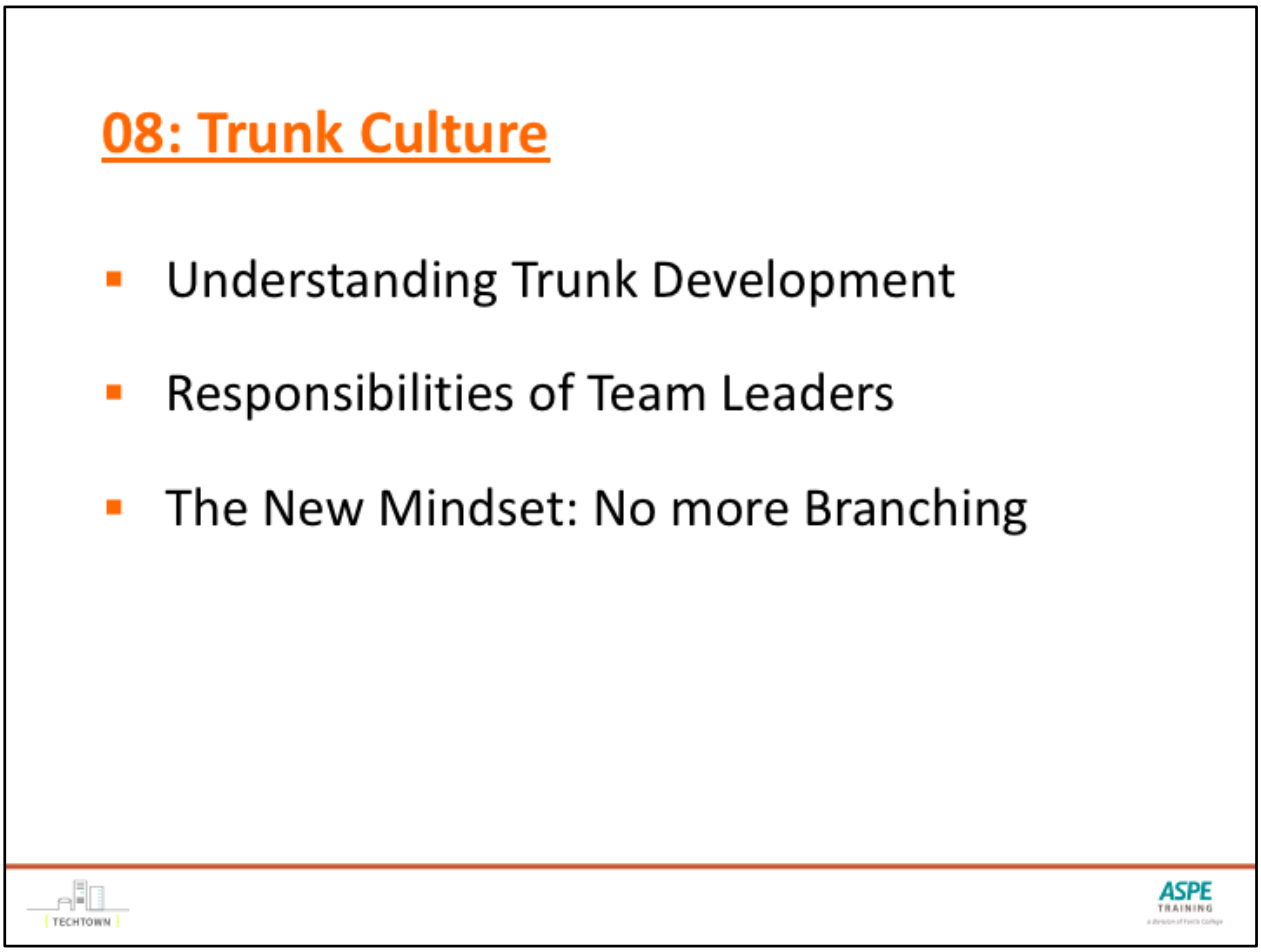
9. In the corresponding code window, scroll down until you see:

```
public ActionResult About()
{
ViewBag.Message = “Your application description”
return View();
}
public ActionResult Contact()
{
ViewBag.Message = “Your contact page”;
```

10. Change **“Your application description”** to “**Service reservation description**”
11. Change “**Your contact page**” to “**Service reservation contact page**”
12. In the **Solution Explorer**, right click on Solution **“Service Reservations WebUI”**
13. Then select the **Check In** menu item, and click the **Yes** button when the dialogue pops up
14. The **Gated Check** in dialogue pops up, accept the defaults and click the Build Changes button
15. In the **Solution Explorer**, click on the link “**Your check-in has been queued for validation. Click here to see the status of the validation build.”**
16. Click on the **Build Summary** link to view the results. Notice our Tests have failed and our **Gated check-in** has been rejected.
17. Review the Test Results in the **Build summary**
18. The build completes, notice that our Code coverage summary shows a very low percentage**.**

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

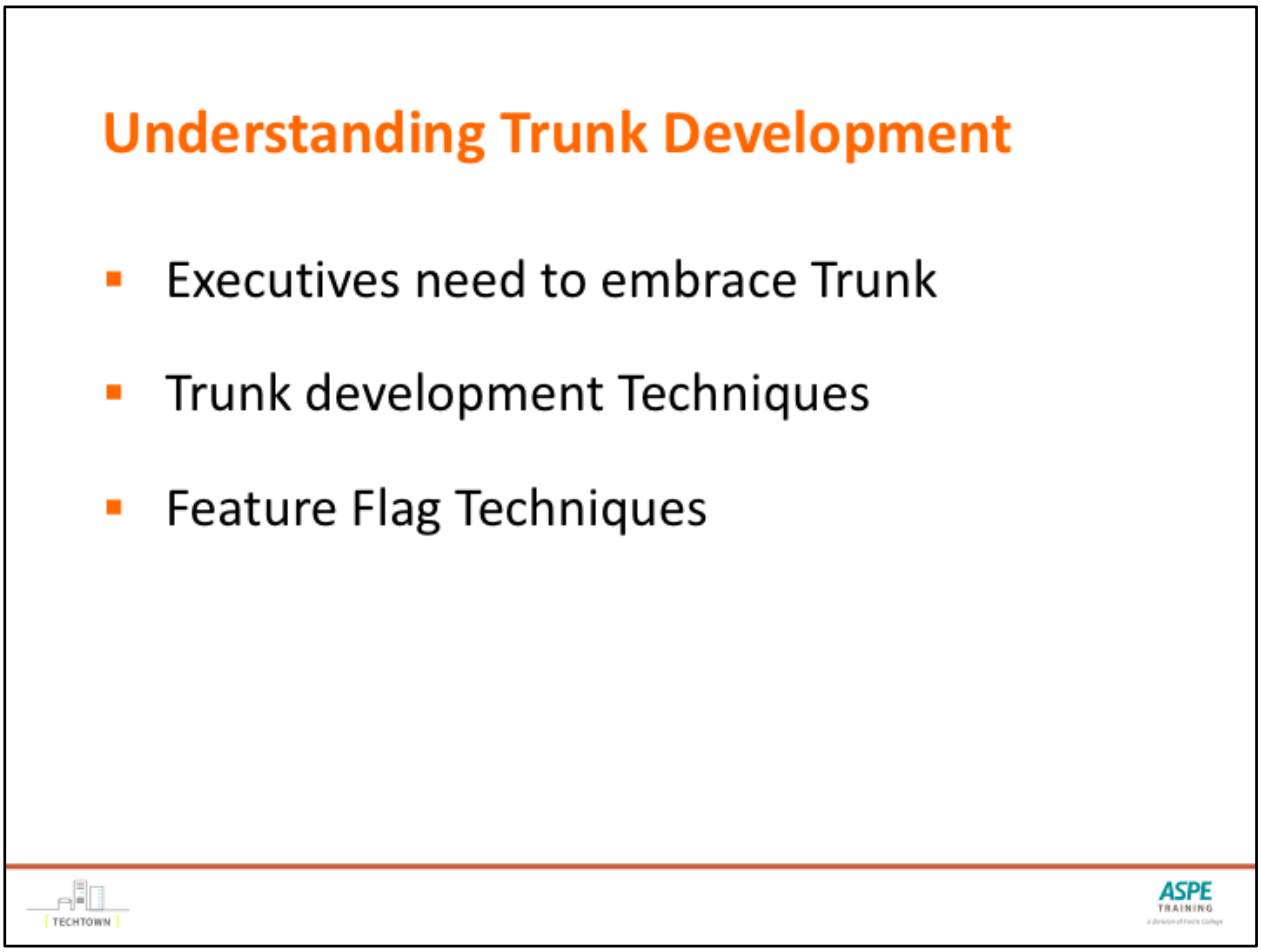


Trunk development Techniques
Cultural transformation is paramount, but technical aspects are equally as important in delivering value to your customer. Here we will discuss some of the techniques that can make the transition more effective.

Executives need to embrace Trunk
Traditionaly larger organizations have been working on the other side of Trunk Development, making it hard to truly understand it's benefits. They will need to be at the forefront of cultural change for the company, so understanding and embracing the upsides will be necessary.

Feature Flag Techniques
Turn features on and off. Pretty much a requirement for mainline development… nor more branching. Feature flags can be deployed "Off" then turned on via the feature flag, they let you manage the entire lifecycle of a feature manage --- all the merge problems go away and it reduces deployment risk

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Executives need to embrace Trunk
Traditionally leaders in larger organizations have been working on the other side of Trunk Development, making it hard to truly understand its benefits. They will need to be at the forefront of cultural change for the company, so understanding and embracing code base priorities is critical:

- The importance of trunk development
- Increasing code stability
- Getting a team to apply DevOps

Trunk Development Techniques
Cultural transformation is paramount, but technical aspects are equally as important in delivering value to your customer. Here we will discuss some of the techniques that can make the transition more effective.

Feature Flag Techniques
Turn features on and off. Pretty much a requirement for mainline development… nor more branching. Feature flags can be deployed "Off" then turned on via the feature flag, they let you manage the entire lifecycle of a feature --- all the merge problems go away and it reduces deployment risk

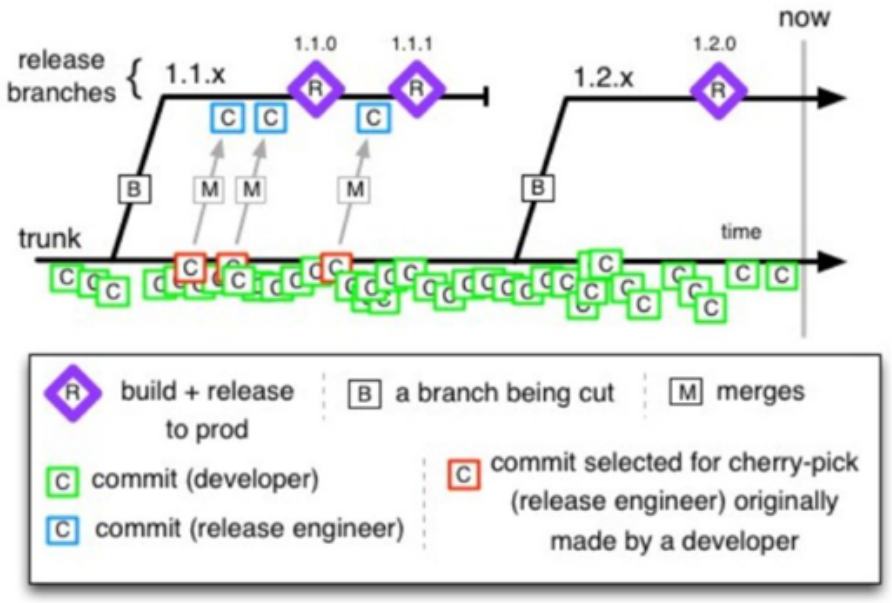
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Trunk Development

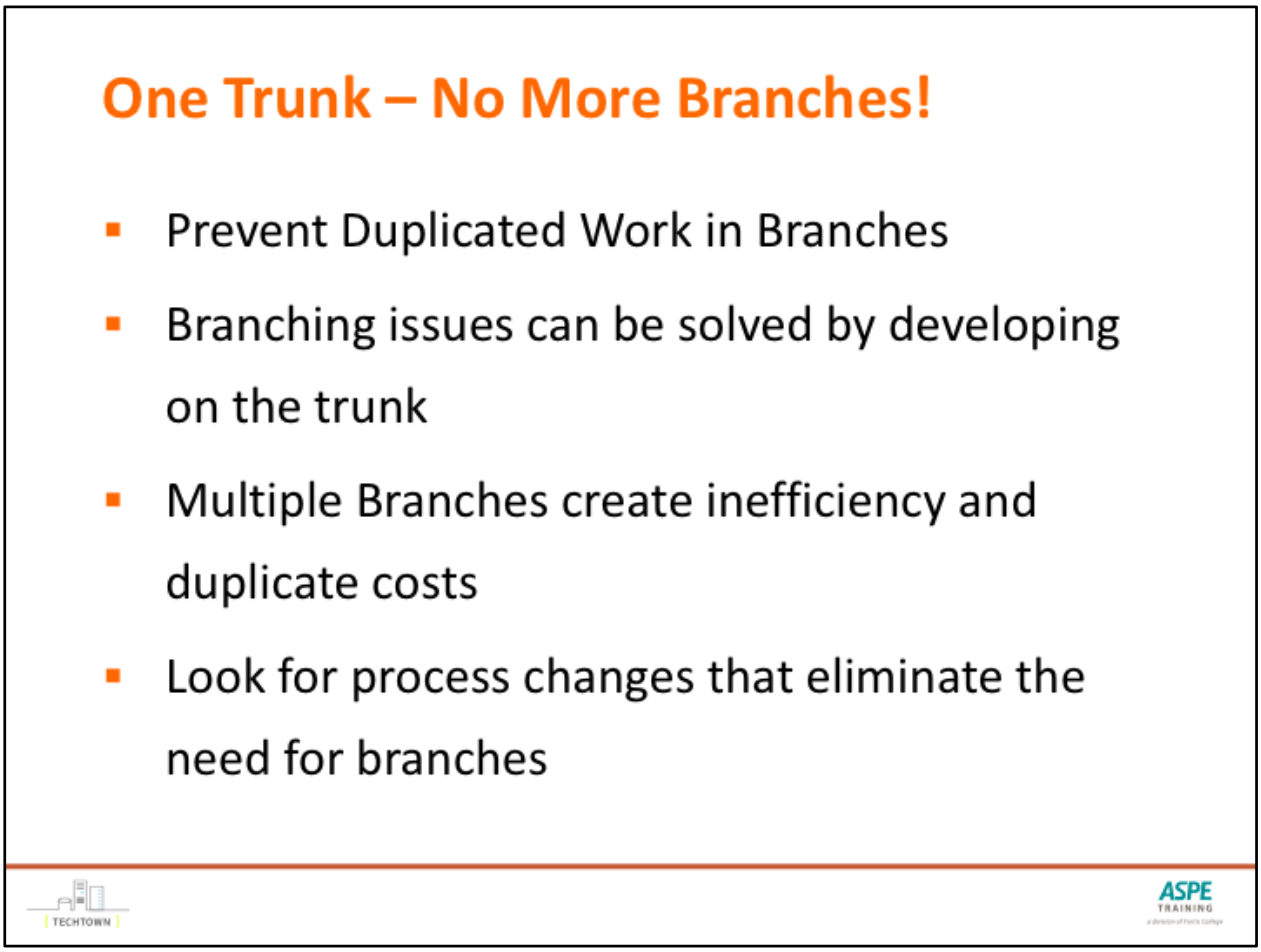
- All developers work on a **single branch**
- Often it's the **master** branch. They commit code to it and run it
- In some cases, they create **short-lived feature branches**. Once code on their branch compiles and passess all tests, they merge it straight to **master**
- Ensures that development is continuous and prevents developers from creating merge conflicts that are difficult to resolve

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Trunk Development



Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Testing Branches is expensive
Managing branches that work with more than one branch of similar code takes time, and negatively affects team capacity.

- Working with Branches creates redundancy of Code
- Testing associated branches for errors is time consuming

Taking Duplication out of the Process
Manual testing is often necessary when working with branches of code. Even though there are compelling arguments to use them, these are duplicate costs. They often end up needing manual tests, making it expensive and time consuming to maintain each branch.

Reasons you will hear for needing branches

- Customer Driven reasons for Branching
- Architectural reasons for Branching

One Trunk – no more branches!
Data shows that developing on one trunk reduces redundancy. It is also often a subject of resistance, because teams are accustomed to working with branches. Use analytics to communicate the benefits of trunk development, and the cultural shift will be justified.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

When does this work and when doesn't

- **When it works best**
  - When you are starting up
  - When you need to iterate quickly
  - When you work with more senior developers
- **When it doesn't work**
  - When you have lot of junior developers
  - When you work with large established teams
  - You want strict control over who can commit when

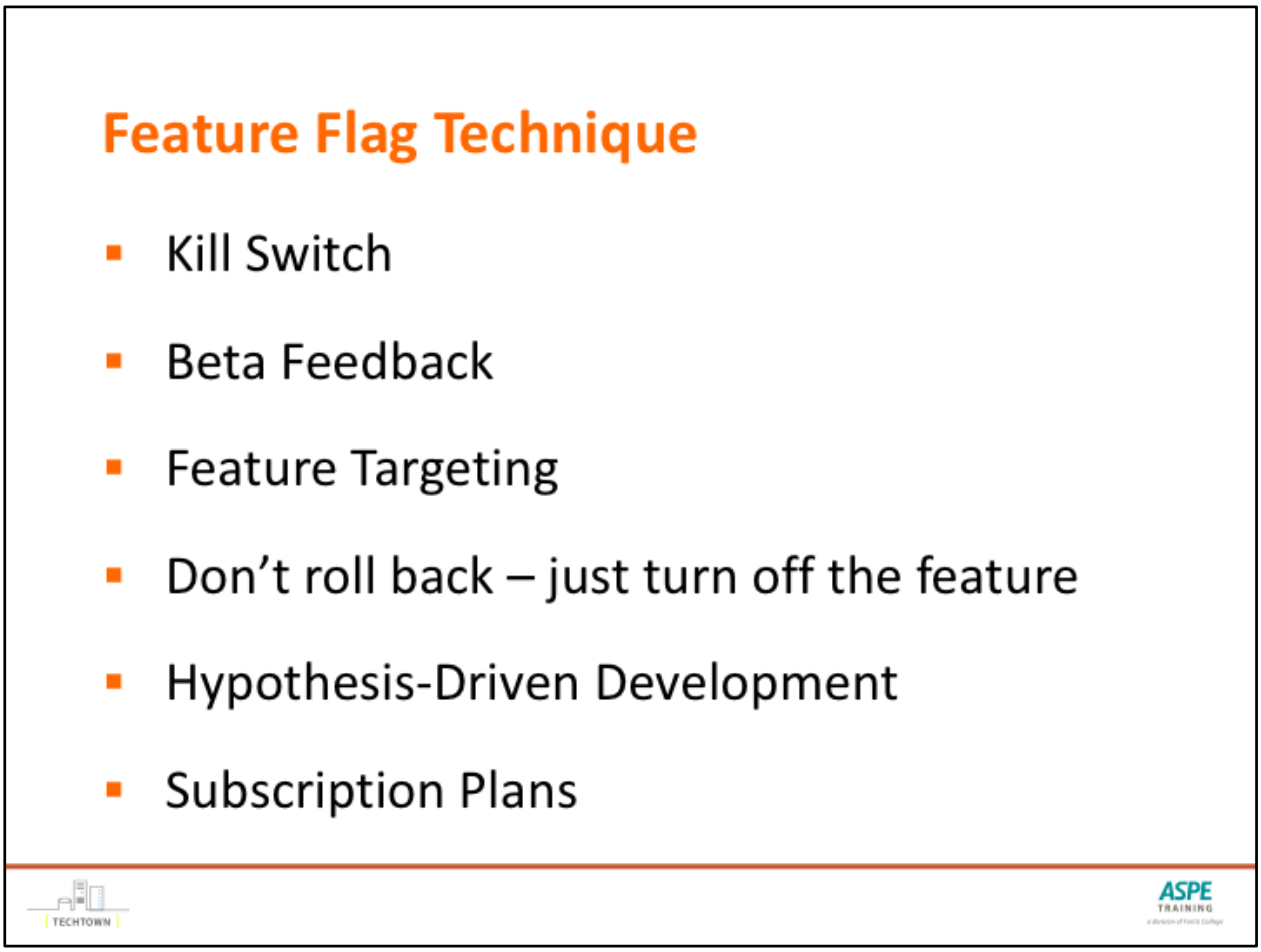
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Trunk Development Techniques

- Versioning and Tagging
- Re-architecture through Abstraction
- Feature Flags

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



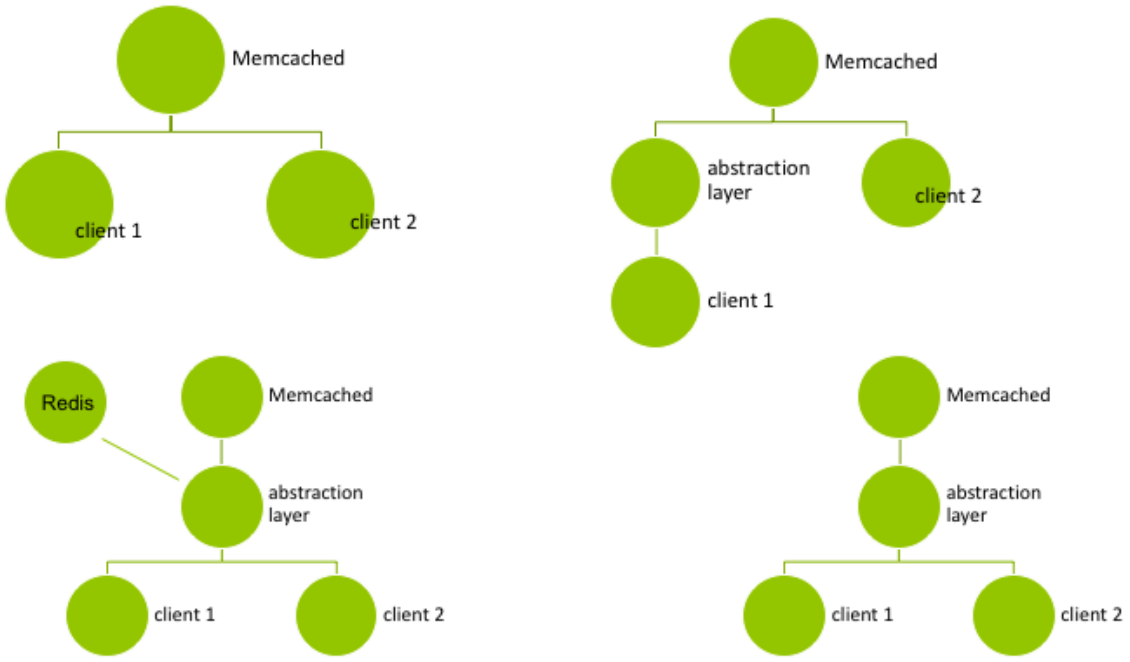
Thoughts on Feature Flags
https://blogs.msdn.microsoft.com/devops/2016/06/24/effective-patterns-for-feature-flags/

Feature Flag Techniques

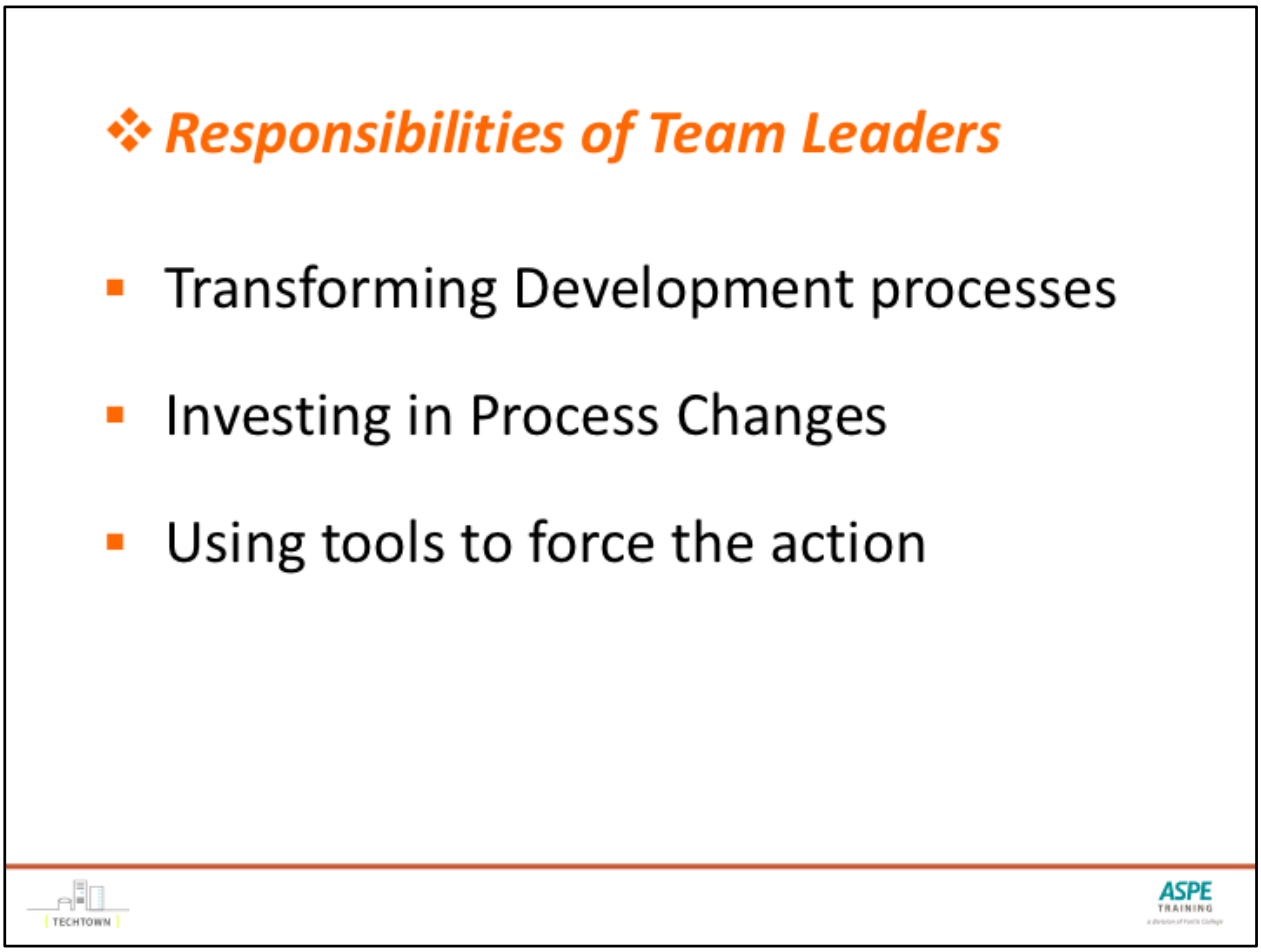
- Using APIs
- Using Enums
- Store in Config files
- Store in Database

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Re-Architecture Through Abstraction Technique



Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Transforming Development processes
Executives will need to help facilitate the necessary shift in dev processes, which means working closely with teams, setting realistic goals, and using data to inform your decisions.

Investing in Process Changes
Technical changes are bound to be resisted, especially by developers, so rules that incorporate them into the cultural shift required for Agile implementation will need to be enforced by leadership. These changes will inevitably take commitment, focus and effort, but the investment of time & energy will be worth it.

Using tools to force the action
Using tools to enforce behavior will help decrease effort needed to keep builds green. Acquiring the appropriate tools should not be taken lightly, and prioritized accordingly.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Transforming Development Processes

- The big Challenge
- Coping with attitude adjustments
- Understanding the objectives

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Investing in Process Changes

- Effective Communication
- Fixing the issue by re-committing code
- Holding Developers accountable for bad code

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Tools to Force the Action

- Utilizing Technology to force the Issue
- Optimizing available tools
- Delivery of new Capabilities

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Resistance to the Trunk
Changing the way people work will always be challenging. Leadership will need to understand the value and stability added by developing on the trunk, and effectively communicate these assets to the development teams.

Real Time Feedback
Developers can get immediate feedback through automated systems like gated check ins, effectively preventing bad code from effecting the rest of the team.

Converting the non-believers
Executives should not expect developers to buy into trunk development until it has been rolled out, they get used to it, and they can see the benefits. Initially, branching will probably be a point of contention, but leadership will need to stick to their guns for as long as it is necessary.

Arguments for Branching
There are many arguments that can and will be made for branching off of the trunk. These will include examples supported by customer value, stability, architectural concerns, and the list goes on.

Shifting Mindsets
The cultural shift required for the implementation of Agile & DevOps is a slow process, that can take months to years, depending on the size of the organization. Leadership needs to be cognizant of this reality, and approach accordingly, preferably in incremental changes that expose value to team members.

Exercise: Identify Deployment Improvements

- **Update your Value Stream Map based to reflect Using Automated Testing**
  - **Estimate new Lead Times and %CAs**
- **Identify how Test Automation improves your Deployment process**

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Telemetry: Metrics, Monitoring, Alerting

Telemetry = "an automated communications process by which measurements and other data are collected at remote points and are subsequently transmitted to receiving equipment for monitoring."

Event = a change in the state of something that is relevant

Metric = data that are captured because they are useful for detecting Events and understanding current state and history

Monitoring = watching metrics to identify Events that are of interest

Alerting = Notifying people when an Event requires action

Refer to Chapter 14 of The DevOps Handbook.

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Five Telemetry "must haves"

- Instrumentation of applications and infrastructure for data collection
- Storage and easy retrieval
- Data dashboards which are aggregated, formatted and presented in a way suitable for the business and IT staff
- Alerts and notifications for event priorities
- Remediation triggers and/or automation

ASPE TRAINING

NOTES:

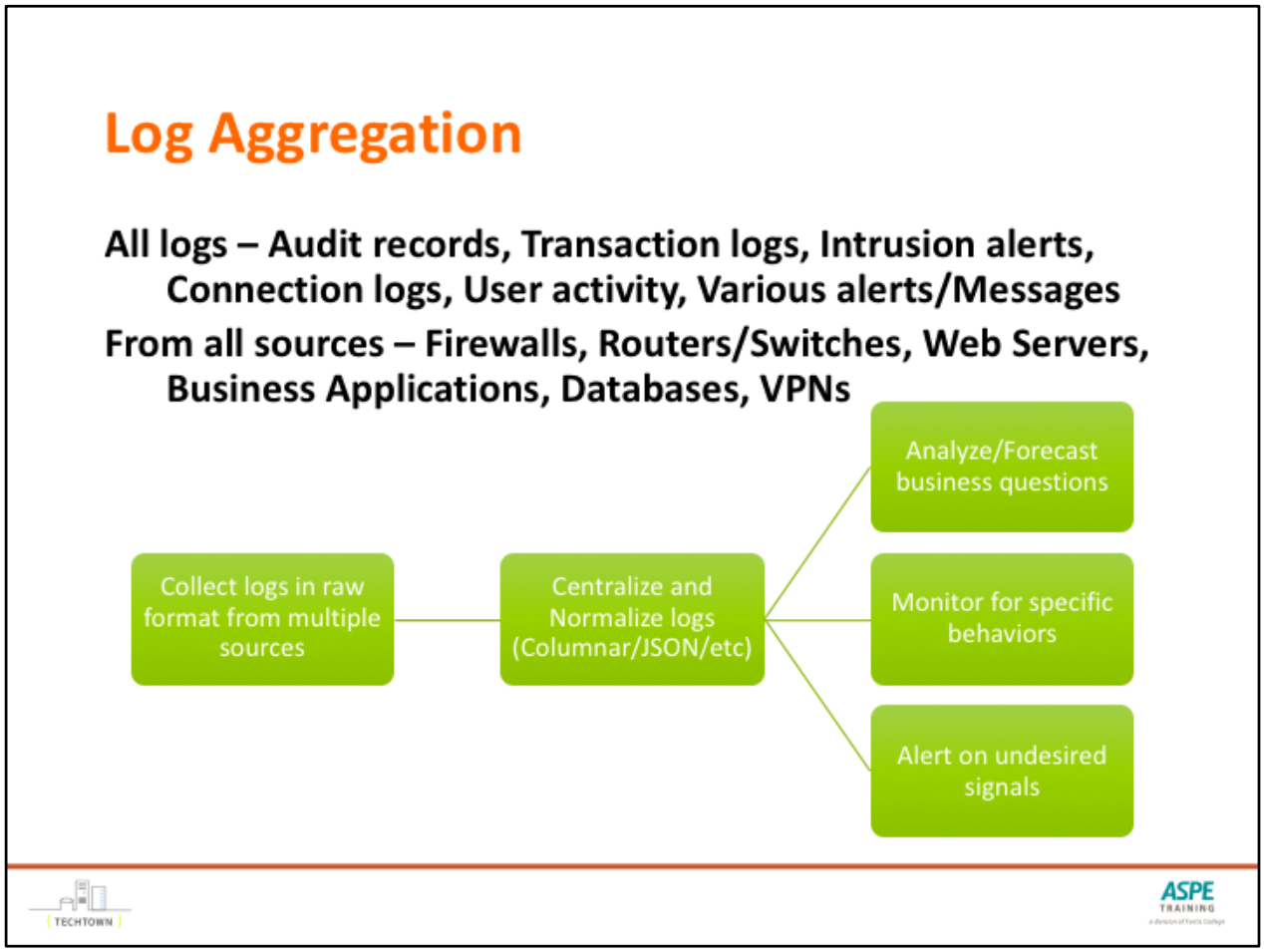
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Telemetry Principles

- Automate Telemetry
- Build Monitoring and Measurement in:
  - Production Environment
  - Development & Test environments
  - Deployment Pipeline
- Use alerts to boost efficiency
- Make information visible to all
  - Within IT and to Customers

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Log management has been around nearly since the beginning of servers. Without a log management strategy, disk space will fill up and data will be forfeited. Lately, some great tools and techniques have emerged to allow for frequent value to be extracted from logs.

One pattern is to centralize or copy logs from node computers to a central storage location and analyze them with a tool such as Syslog. This has an added benefit of allowing for logs of multiple different hosts and applications to be potentially combined and analyzed for complex behavior such as application transactions or even intrusion detection. Additionally, compression tools such as Logrotate can extend the life of a centralized logging strategy.

Once logs are being properly centralized and compressed, a growing selection of tools are available to analyze and present data from them.

There is a new pattern emerging for log analysis, which instead of collecting the logs to a central location chooses to instead analyze the logs in place. The advantages of this approach include the benefits of scaling as there is no theoretical limit on space if the logs are analyzed where they are generated in a distributed manner. This is part of the intention behind tools such as Google's Dremel and Apache Drill.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Common DevOps Metrics

- Number and frequency of software releases
- Volume of defects
- Time/cost per release
- MTTR (Mean Time to Recover)
- Number and frequency of outages / performance issues
- Revenue/profit impact of outages / performance issues
- Number and cost of resources

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Telemetry Tool Set

No one tool is sufficient for Telemetry

- Data Collection
- Data Storage
- Data Aggregation
- Data Retrieval
- Data Visualization
- Dashboards

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

System Monitoring Tools

- Librato
- Monit
- Nagios
- PagerDuty

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Log Aggregation Tools

- Splunk
- ELK

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Visualization Tools

- Kibana
- Grafana

NOTES:

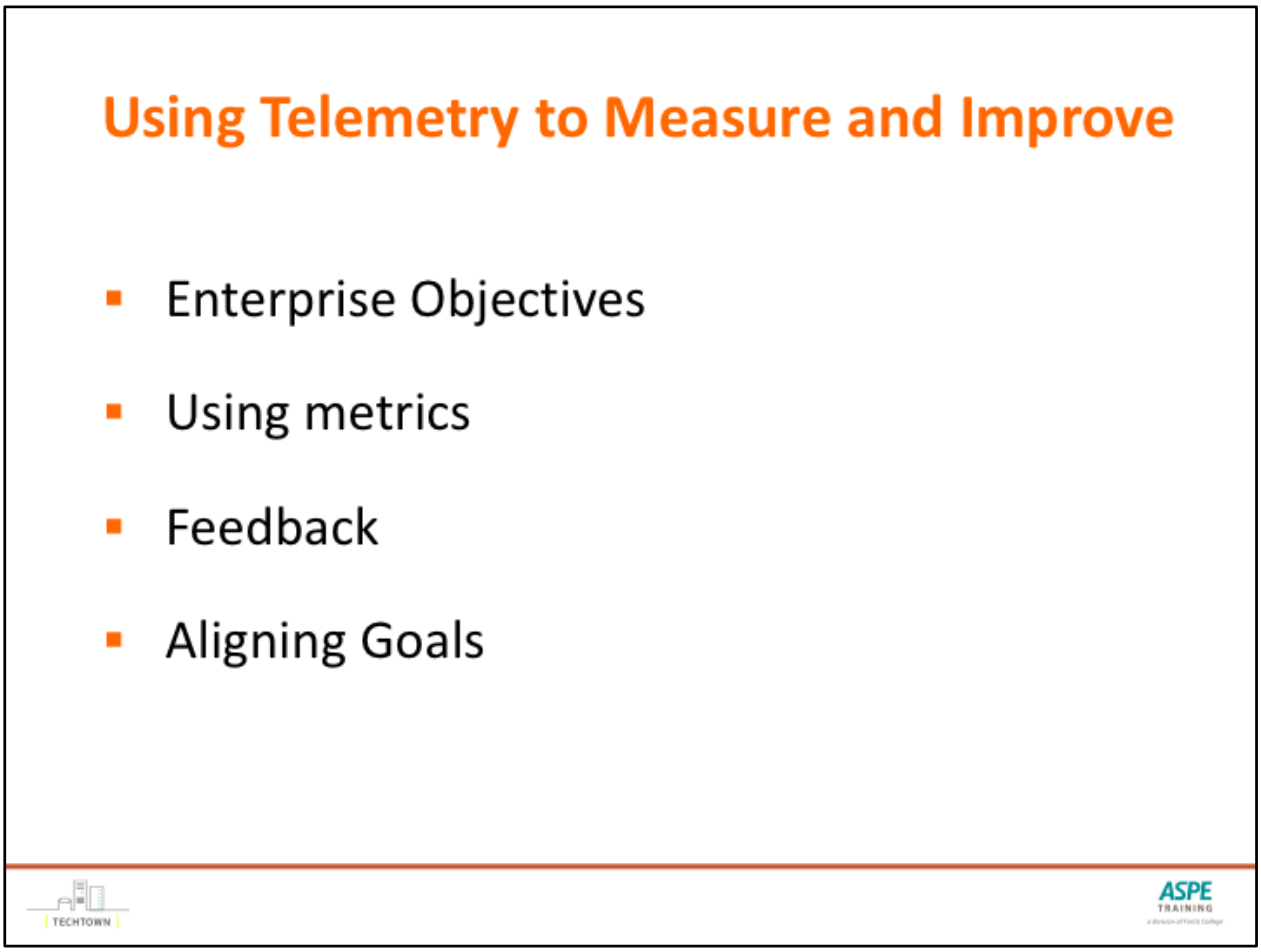
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise: Identify Deployment Logging

- Identify what data should be logged in your Deployment process

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



If a company wants to facilitate continuous improvement, higher level goals may not be in sync with objectives of the developers & testers on the front line. The organization will need to track metrics, identify challenges, and engage with everyone involved to prevent bottlenecks and defects. This chapter focuses on various techniques that can be used to implement business-specific solutions that address these challenges, and facilitate agreement from the top down, and the bottom up.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Data to Identify Challenges

- Using metrics to identify goals
- Utilizing Objectives to guide the work
- Where are developers struggling, and why?

ASPE TRAINING

NOTES:

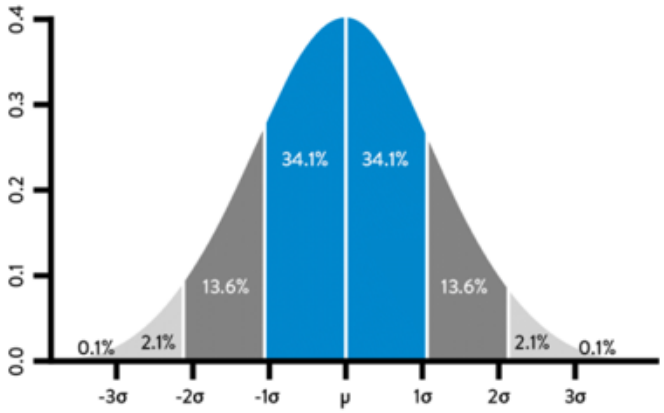
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Use Telemetry To Anticipate Problems

Use statistical techniques to understand “normal”, then address the “abnormal”

- e.g. Mean and Standard Deviation

Example: NetFlix



Refer to Chapter 15 of The DevOps Handbook.

NetFlix Example:
Roy Rapoport: “Given a herd of cattle that should all look and act the same, which cattle look different from the rest? Or more concretely, if we have a thousand-node stateless compute cluster, all running the same software and subject to the same approximate traffic load, our challenge is to find any nodes that don’t look like the rest of the nodes. Netflix used outlier detection in a very simple way, which was to first compute what was the ‘current normal’ right now, given population of nodes in a compute cluster. And then we identified which nodes didn’t fit that pattern, and removed those nodes from production. We can automatically flag misbehaving nodes without having to actually define what the ‘proper’ behavior is in any way. And since we’re engineered to run resiliently in the cloud, we don’t tell anyone in Operations to do something— instead, we just kill the sick or misbehaving compute node, and then log it or notify the engineers in whatever form they want. Netflix has massively reduced the effort of finding sick servers, and, more importantly, massively reduced the time require to fix them, resulting in improved service quality. The benefit of using these techniques to preserve employee sanity, work/ life balance, and service quality cannot be overstated.”

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Use Telemetry To Anticipate Problems

Monitor for (and alert on) trends.. e.g.:

- Increasing web page load times
- Decreasing server free memory
- Decreasing available disk space
- Growing database transaction times
- Declining number of functioning servers behind the load balancer

ASPE TRAINING

NOTES:

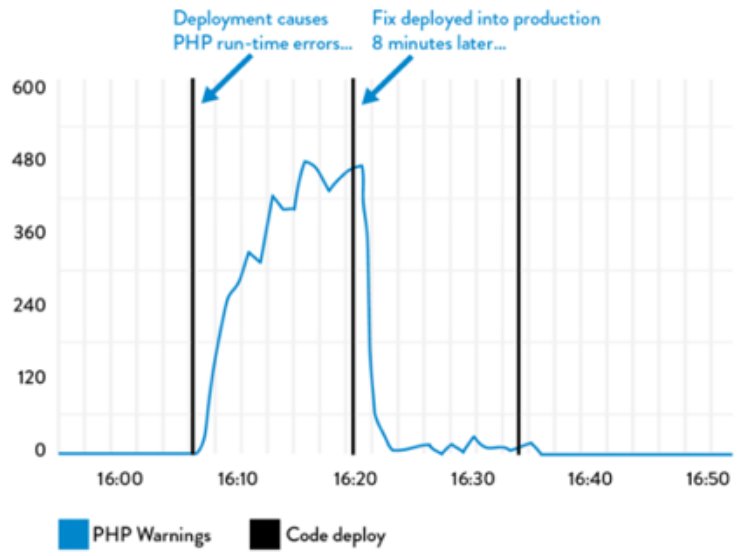
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Use Telemetry for Safe Deployment of Code

Actively Monitor key metrics immediately before and after each deploy

Take action on unexpected changes

Example: Etsy.com



Refer to Chapter 16 of The DevOps Handbook.

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Use Telemetry to understand Issues at the Executive Level

- Communicating Challenges to Executives
- Executives are there to help
- Executive participation in the cultural shift

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Telemetry Metrics to Increase Progress

- A new role for executives
- Understanding which Processes work
- Learning what needs improvement

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Telemetry Metrics to Identify Challenges

- Sharing ideas for improvement
- Engineers, Developers, and Executives
- Trust enables a greater ability to fix issues

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Telemetry and Metrics to Improve Stability

- What are the teams achieving?
- Integrating success into future iterations
- Adjusting processes to prevent errors

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Customer-Oriented Telemetry

Hear the "Voice of the Customer"

- Applications capture:
  - Customer activities
  - User actions and mistakes
  - Business activity

Discuss: Customer-Oriented telemetry

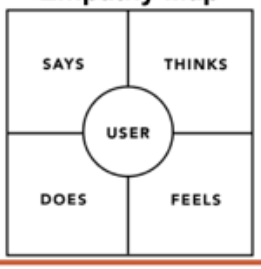
- What data would provide insights?

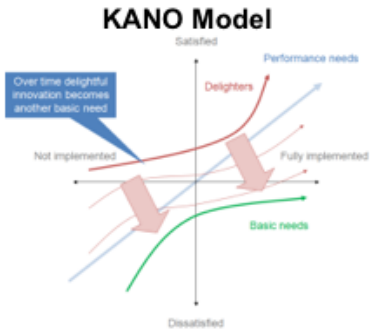
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Using Telemetry to Understand the Voice of the Customer





TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Discussion
Voice of the Customer

- What telemetry can be useful to understand the Voice of the Customer?
- How could you use it?

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Exercise: Identify Telemetry to be Reported, Displayed, or Alerted

- Identify what telemetry should be reported, displayed or alerted on in your Deployment process
- Determine if any sensitive information or PII is being (or should be) logged. If so determine what to do about it. (e.g. stop logging it, or protect it)

TECHTOWN

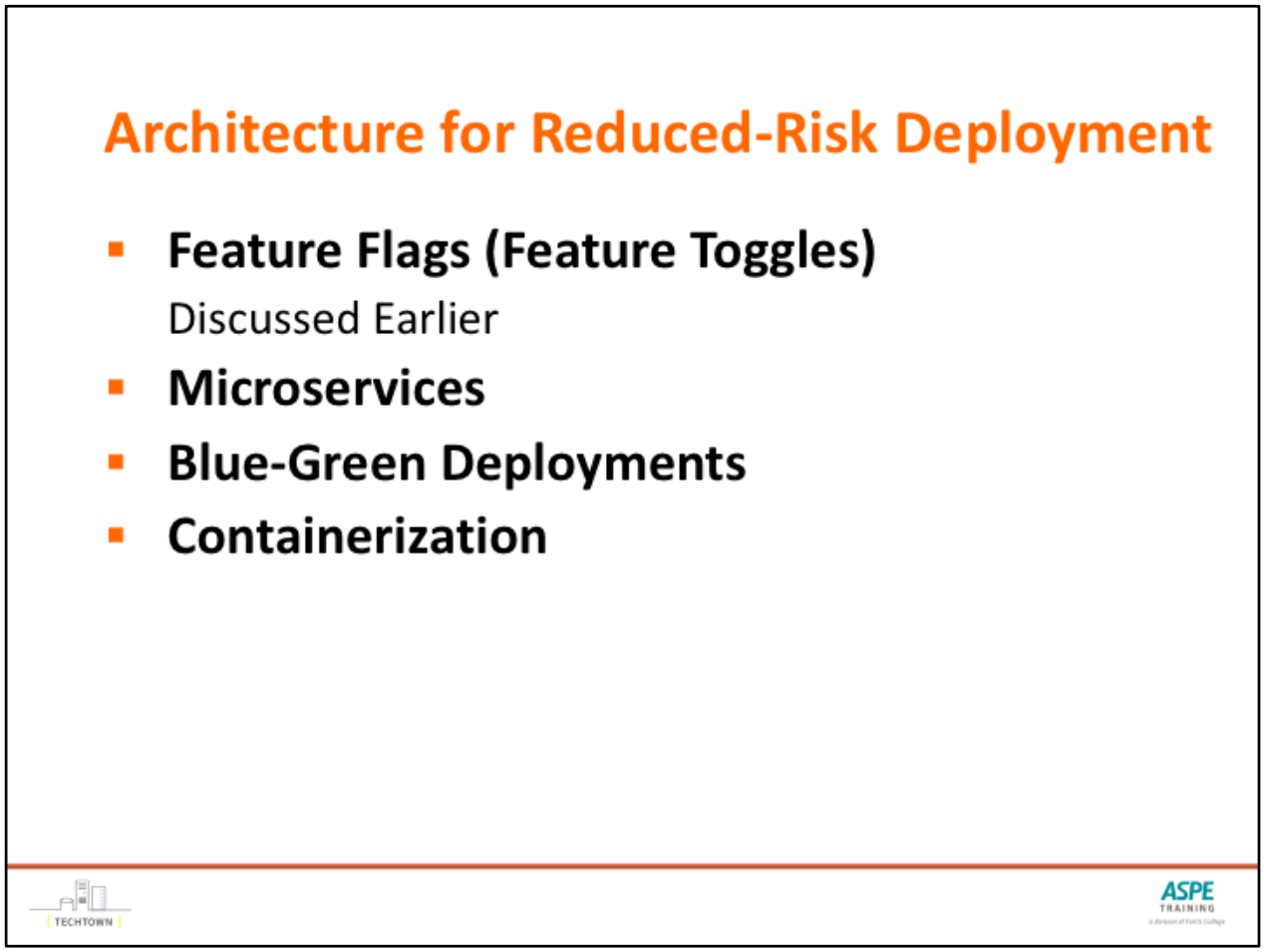
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

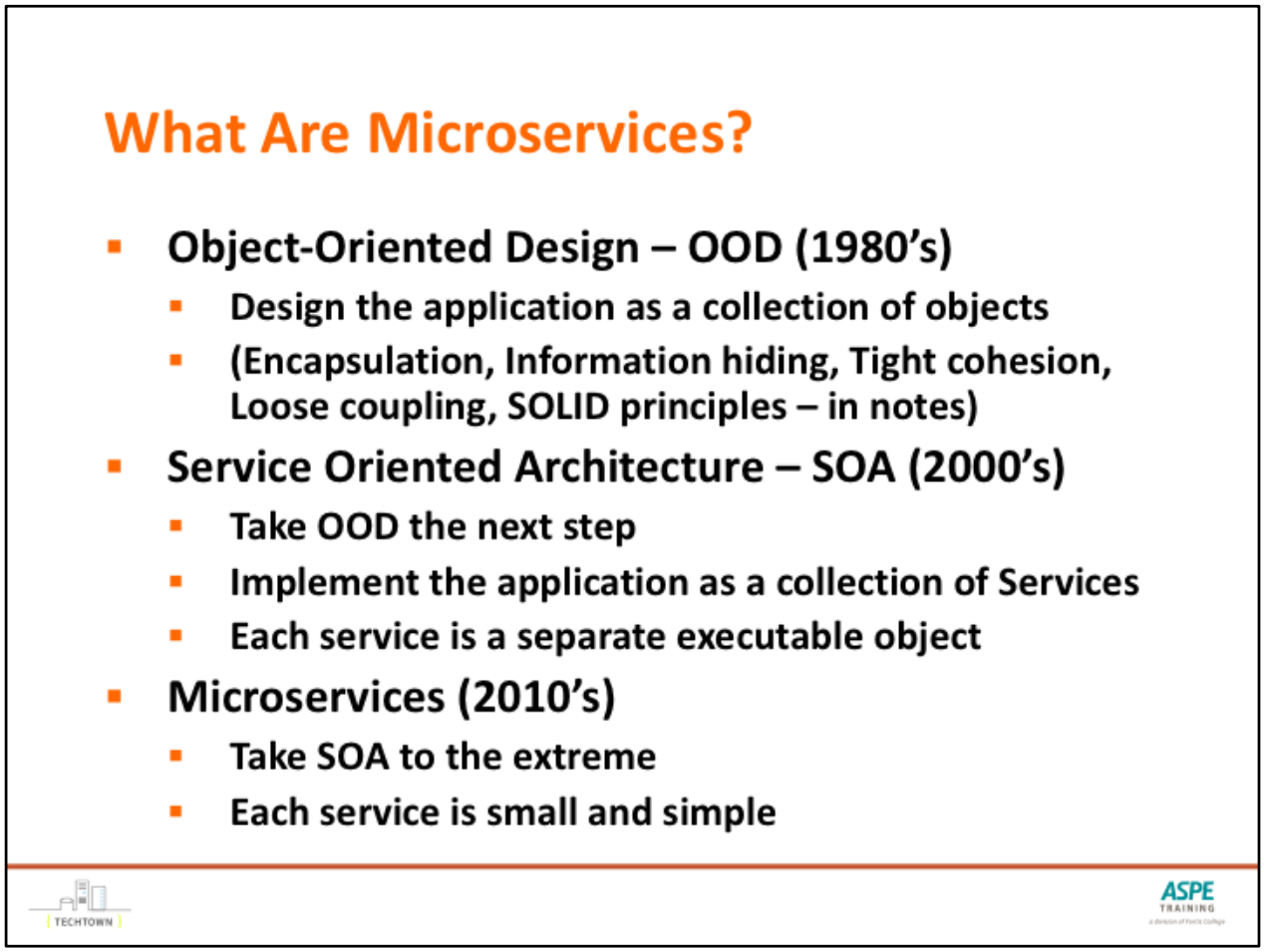
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Refer to Chapter 13 of The DevOps Handbook.

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



S – Single Responsibility – A given method should only be responsible for a single axis of change.

O – Open / Closed Principle – Objects should be open for extension, but closed for modification.

L – Liskov Substitution – Derived types should be completely substitutable for their base types.

I – Interface Segregation – Favor many, specific interfaces over a single "fat" interface. An API should only contain the methods its caller needs to use.

D – Dependency Inversion – If you want an easier life as a coder, "depend on abstractions, not concretions", that is, decouple any dependencies through well-placed abstraction.

https://quickleft.com/blog/solid-design-principles/

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Why Use Microservices?

- Overall system resilience
  - Service failure does not kill the application
  - Service failure may be invisible to users
- Scalability
  - Replicate a Microservice to accommodate demand
- Observe Conway's Law
  - Small Agile Teams & Microservices

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

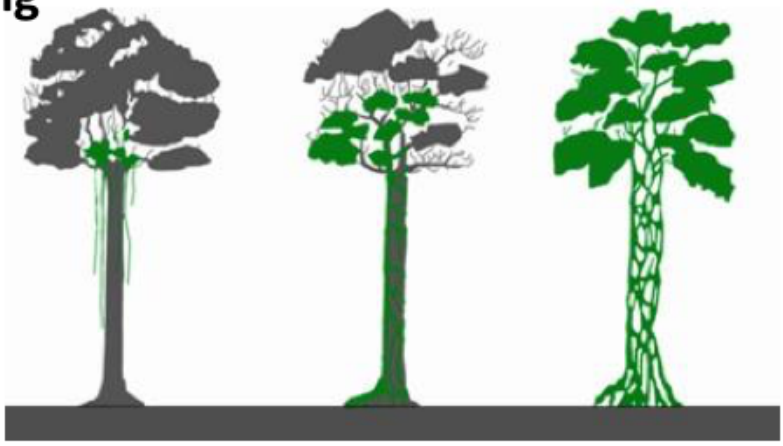
Re-architecting Applications Gradually
The Strangler Pattern

The 'Strangler' Pattern, as described by Martin Fowler. The pattern (based on the Strangler Fig – pictured above) involves gradual replacement of a system by implementing new features in a new application that is loosely coupled to an existing system. Porting existing functionality from the original application only where necessary.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Why The Strangler Pattern?

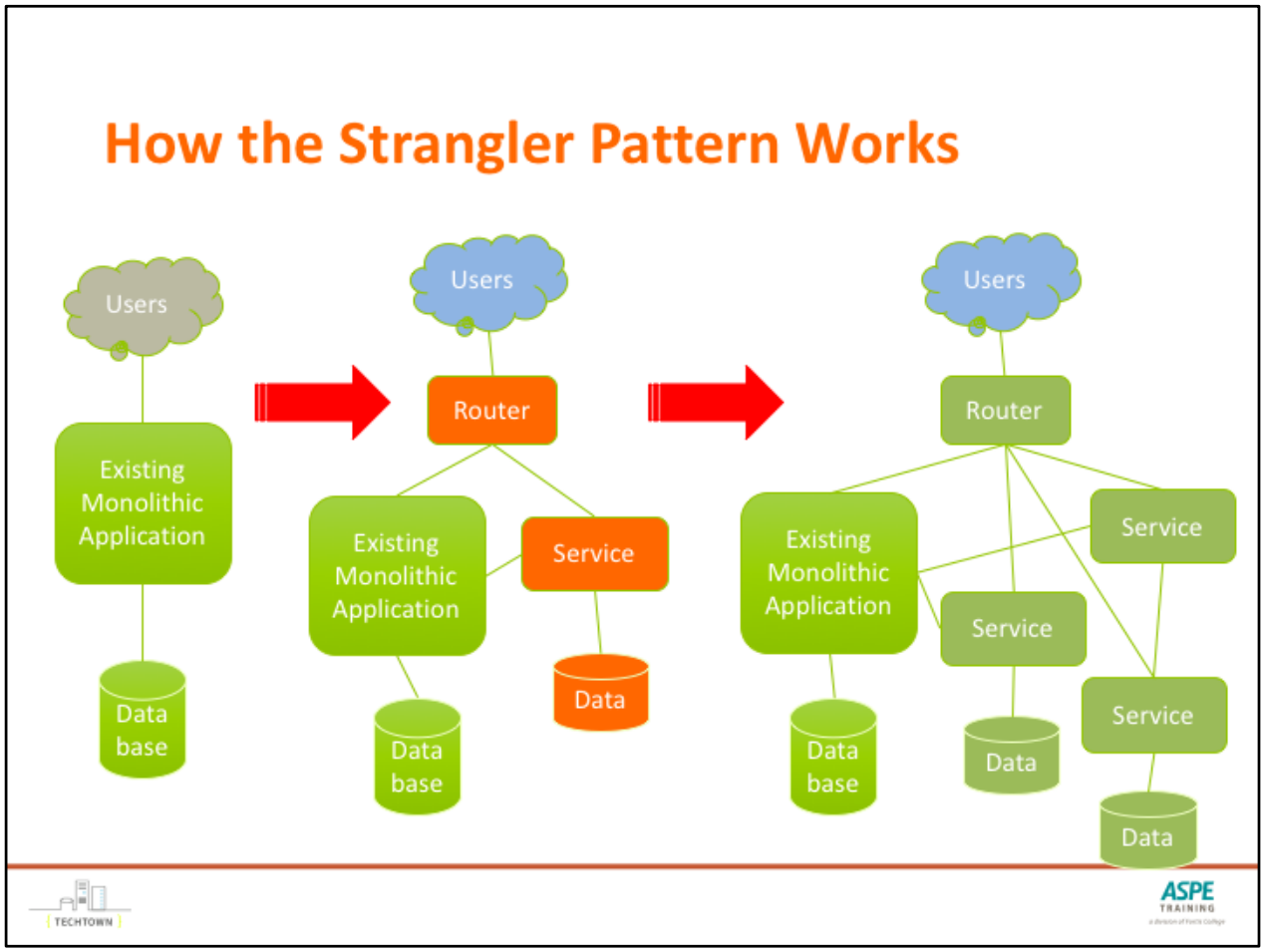
- Avoid risky application rewrite projects
- Start seeing the benefits of SOA quickly
- Begin moving toward smaller market-oriented teams



One big benefit of this pattern is that is much more likely to get approved. Enterprises will debate replacing a large legacy system for years, but a team can deliver a quick prototype that replaces a single feature in a sprint. This effort can be built upon and a difficult legacy software component can be replaced in a more iterative and natural manner.

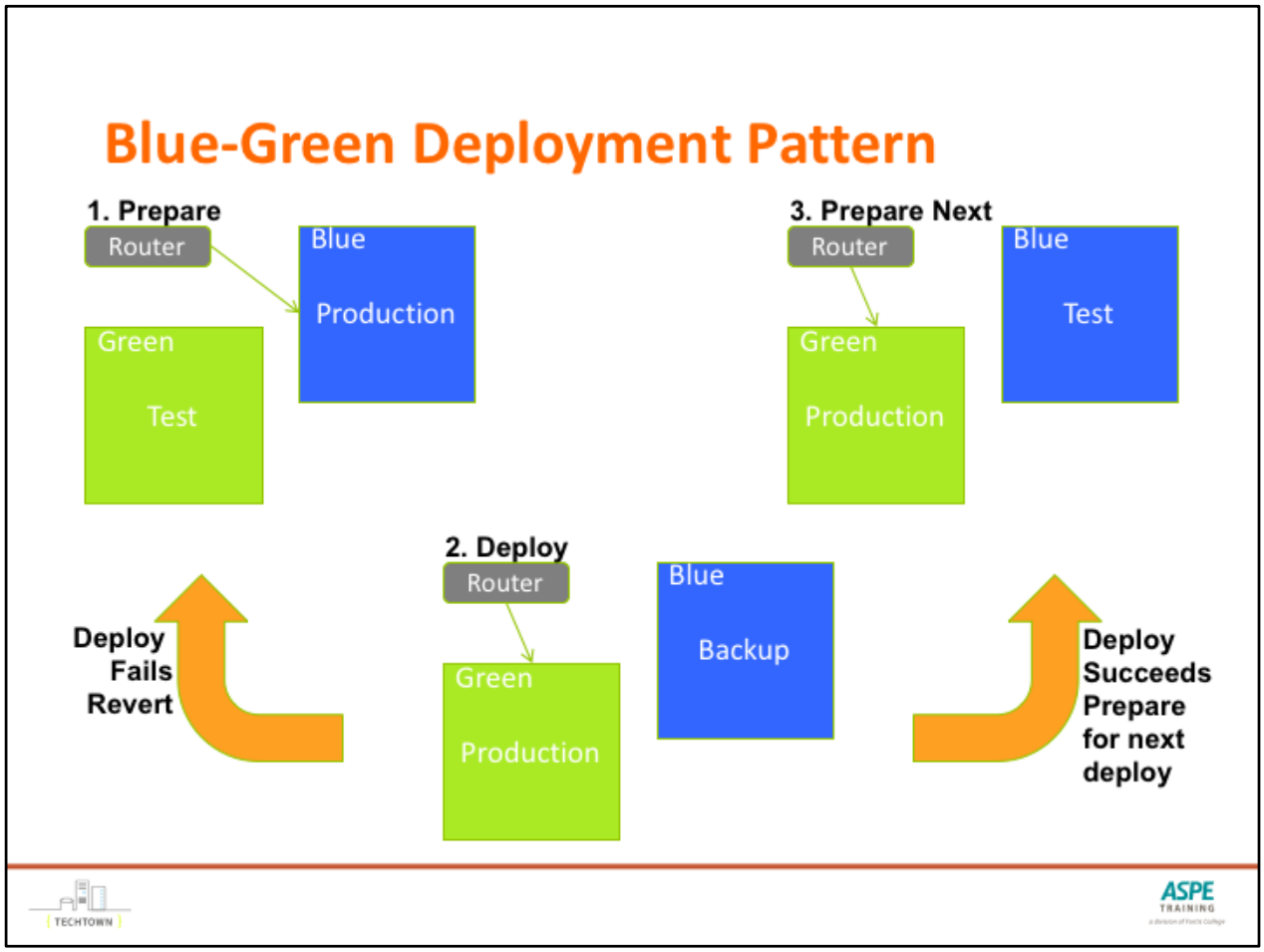
Another benefit is the deployability of new components. Ideally building them on a PaaS or other rapid development infrastructure.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Instead of making changes to the old application, new or changed functionality is implemented in new micro-services. Over time, more and more of the old application will be abandoned in place until eventually, when none of the original application is being used, it can be turned off.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

What is Docker?

The Docker Project

Open Source Project

- 2B+ Docker Image Downloads
- 2000+ contributors
- 40K+ GitHub stars
- 200K+ Dockerized apps
- 240 Meetups in 70 countries
- 95K Meetup members

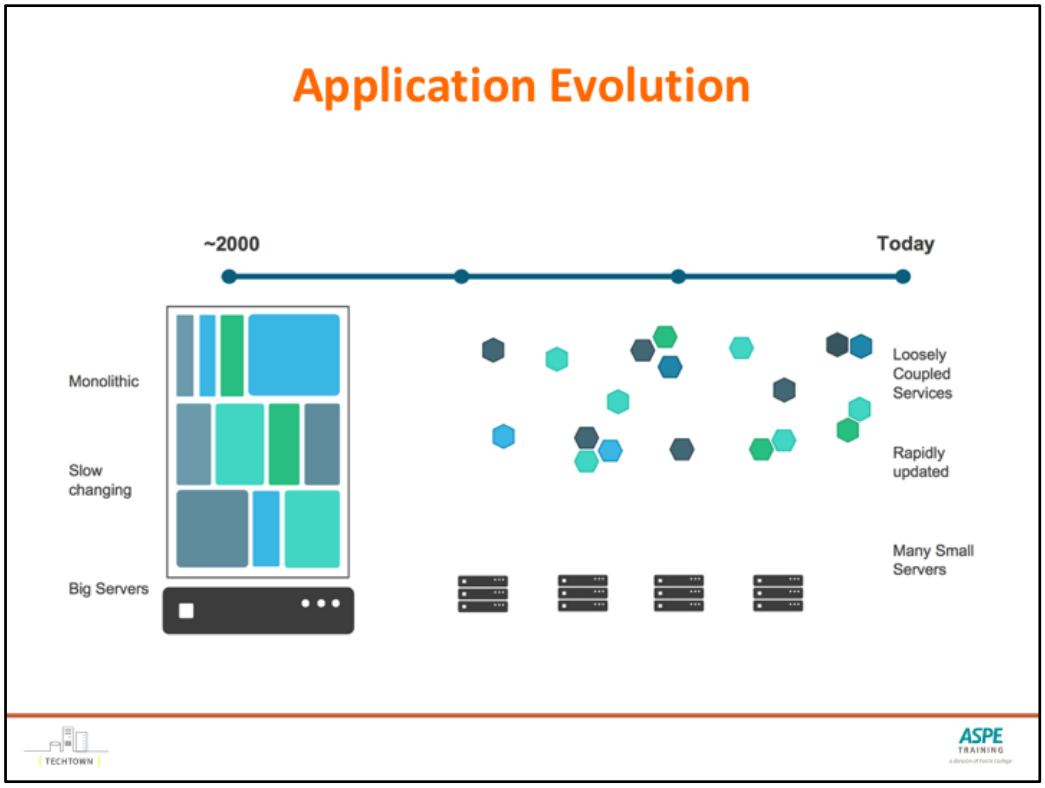
Docker Inc

Containers as a Service provider

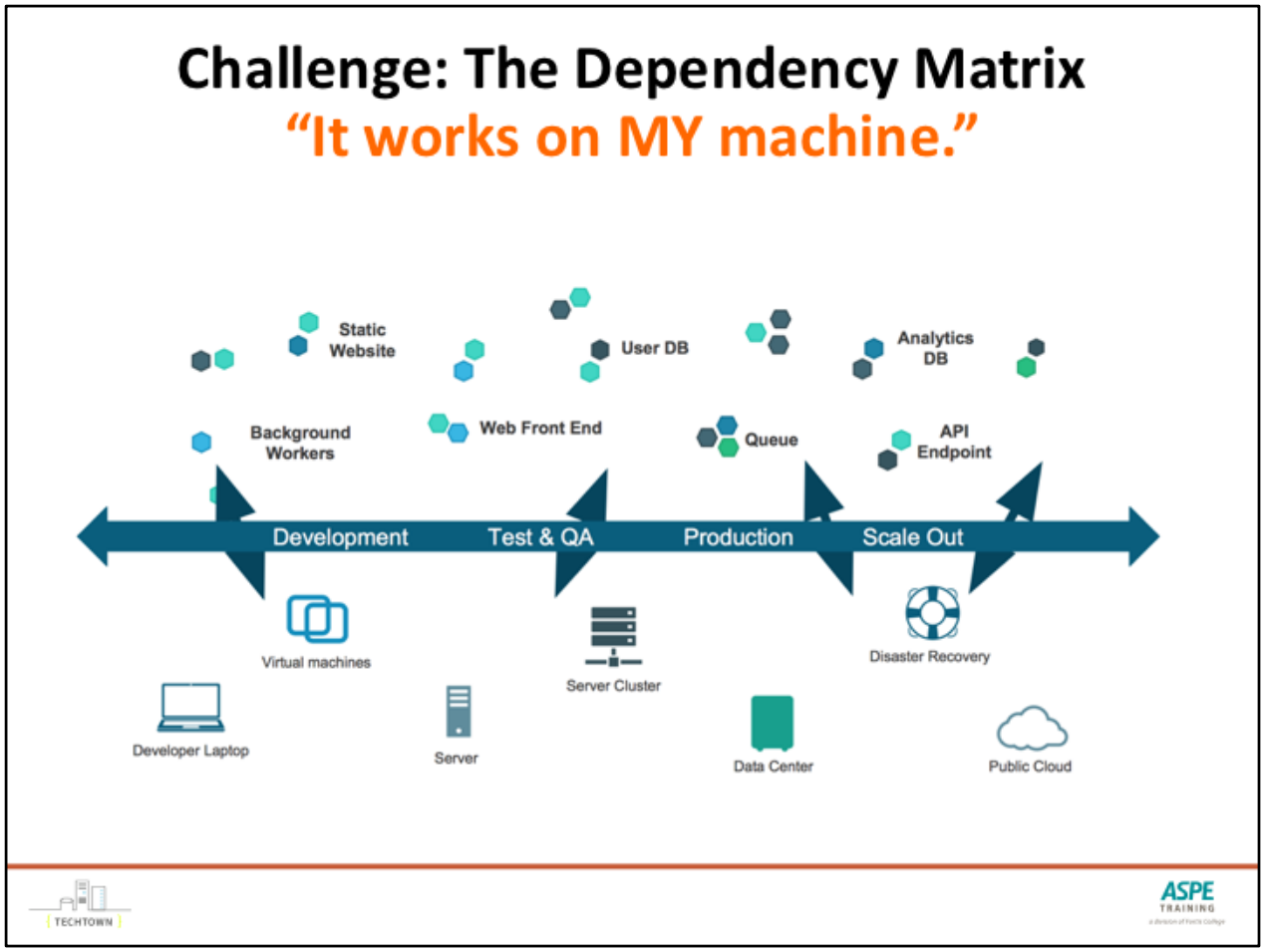
- Integrated platform for dev and IT
- Commercial technical support

Docker project sponsor

- Primary sponsor of Docker project
- Supports project maintainers



- Management of software has changed radically in this time too:
  - We used to have big up-front specifications for “projects” that represented yearly releases, and deadlines were frequently missed.
  - Now we have Agile development… “projects” are mostly gone and instead we have ever-changing “products” with a much-tighter release schedule… sometimes multiple times per day.



A common problem for developers is the difficulty of managing all their application's dependencies in a simple and automated way.
This is usually difficult for several reasons:

Cross-platform dependencies. Modern applications often depend on a combination of system libraries and binaries, language-specific packages, framework-specific modules, internal components developed for another project, etc. These dependencies live in different "worlds" and require different tools - these tools typically don't work well with each other, requiring awkward custom integrations.

Conflicting dependencies. Different applications may depend on different versions of the same dependency. Packaging tools handle these situations with various degrees of ease - but they all handle them in different and incompatible ways, which again forces the developer to do extra work.

Custom dependencies. A developer may need to prepare a custom version of their application's dependency. Some packaging systems can handle custom versions of a

dependency, others can't - and all of them handle it differently.

Docker solves the problem of dependency hell by giving the developer a simple way to express *all* their application's dependencies in one place, while streamlining the process of assembling them. If this makes you think of XKCD 927, don't worry. Docker doesn't *replace* your favorite packaging systems. It simply orchestrates their use in a simple and repeatable way. How does it do that? With layers.

Docker defines a build as running a sequence of Unix commands, one after the other, in the same container. Build commands modify the contents of the container (usually by installing new files on the filesystem), the next command modifies it some more, etc. Since each build command inherits the result of the previous commands, the *order* in which the commands are executed expresses *dependencies*.

Here's a typical Docker build process:

FROM ubuntu:12.04 RUN apt-get update && apt-get install -y python python-pip curl RUN curl -sSL https://github.com/shykes/helloflask/archive/master.tar.gz | tar -xzv RUN cd helloflask-master && pip install -r requirements.txt

Note that Docker doesn't care *how* dependencies are built - as long as they can be built by running a Unix command in a container.

Reference: https://github.com/docker/docker

NOTES:

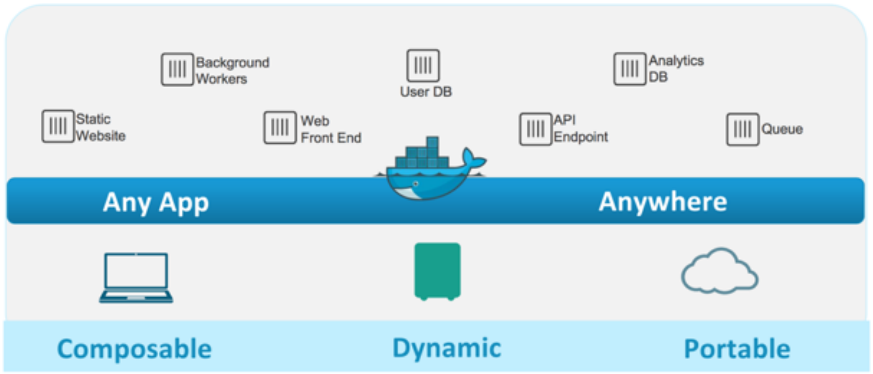
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Containers as a Solution

Container

- Packages up software binaries and dependencies
- Isolates software from each other
- Container is a standard format
- Easily portable across environment
- Allows ecosystem to develop around its standard

Containers as a Solution



Developer Benefits

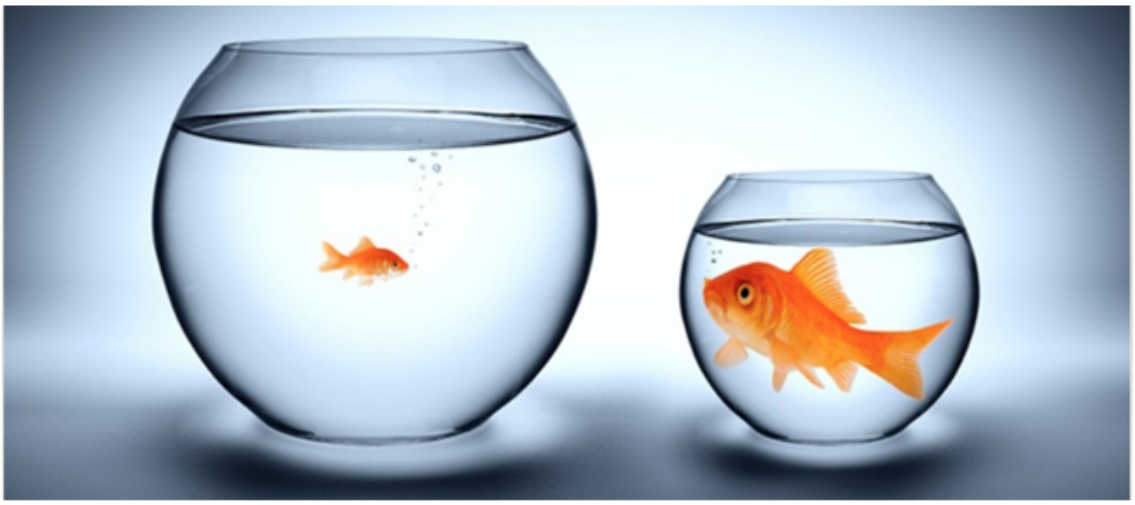
- Build once...(finally) run anywhere
  - A clean, safe, hygienic and portable runtime environment for your app.
  - No worries about missing dependencies, packages and other pain points during subsequent deployments.
  - Run each app in its own isolated container, so you can run various versions of libraries and other dependencies for each app without worrying
  - Automate testing, integration, packaging...anything you can script
  - Reduce/eliminate concerns about compatibility on different platforms, either your own or your customers.
  - Cheap, zero-penalty containers to deploy services? A VM without the overhead of a VM? Instant replay and reset of image snapshots?

One big driver for the initial adoption of Docker has been startups in an effort to drive down their Cloud Bill. There is lots of anecdotal evidence of startups cutting their cloud bill in half by using Docker to increase utilization. Instead of having a single VM for a database for example, putting a database, mid tier and front end container all in the same VM thus using less resources overall.

Operational Benefits

- Configure once...run anything
  - Make the entire lifecycle more efficient, consistent, and repeatable
  - Increase the quality of code produced by developers.
  - Eliminate inconsistencies between development, test, production, and customer environments
  - Support segregation of duties
  - Significantly improves the speed and reliability of continuous deployment and continuous integration systems
  - Because the containers are so lightweight, address significant performance, costs, deployment, and portability issues normally associated with VM

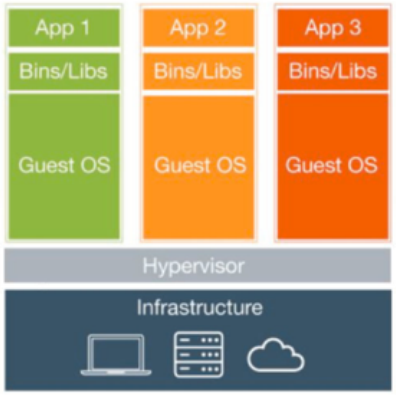
VMs vs. Containers



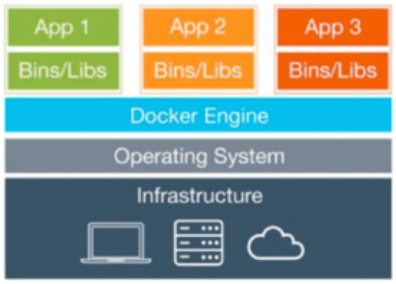
A badly kept secret on cloud is the over-provisioning of VMs for workloads. The average CPU utilization on the average data center VM is between 6-12% and this is part of the reason for a very healthy ecosystem of cloud consultants.
Containers are more lightweight and are typically filled with higher density workloads.

https://gigaom.com/2013/11/30/the-sorry-state-of-server-utilization-and-the-impending-post-hypervisor-era/

VMs vs. Containers



Virtual Machines



Containers

Container Advantages

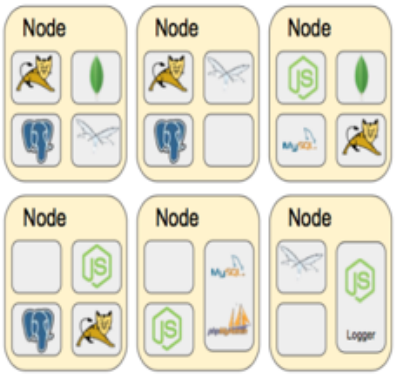
- Portable
- Isolated
- Lighter footprint & overhead (vs VMs)
- Simplify DevOps practices
- Speed up Continuous Integration
- Empower Microservice architectures and adoption

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Challenges with multiple containers



- How to scale?
- Once I scale, where are they?
- How do my containers find each other?
- How should I manage port conflicts?
- What if a host fails?
- How to update them? Health checks?
- How will I track their logs?

TECHTOWN

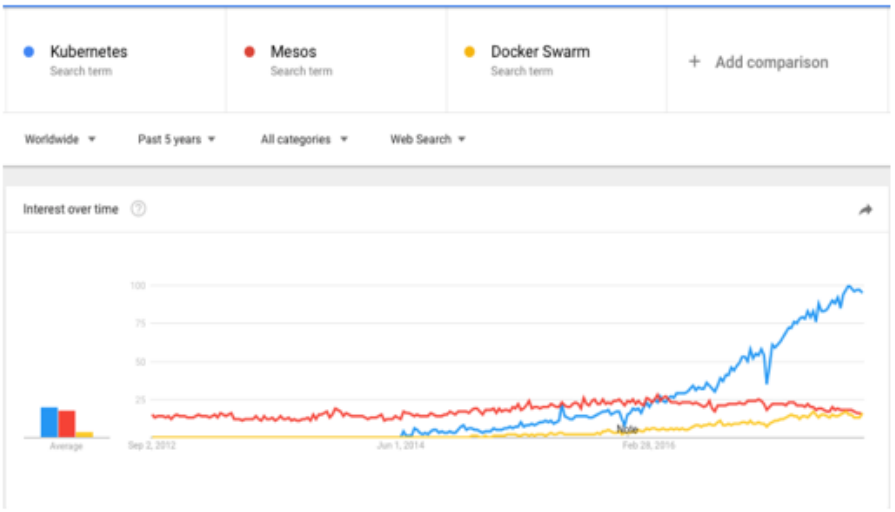
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Container Orchestration

- The three most popular are: Kubernetes, Docker Swam & Mesos
- Kubernetes has become the unofficial standard



TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Orchestration Advantages

- Declarative approach to deploying applications
- Highly scalable, Highly available
- Self Healing
- Tonnes of APIs for container deployment, Jobs, ABAC, RBAC
- Cluster Management and Monitoring
- Automates application configuration through service discovery
- Maintains and tracks the global view of the cluster
- APIs for deployment workflows
  - Rolling updates, canary deploys, and blue-green deployments, scaling, auto-scaling

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Foundations of CI/CD security

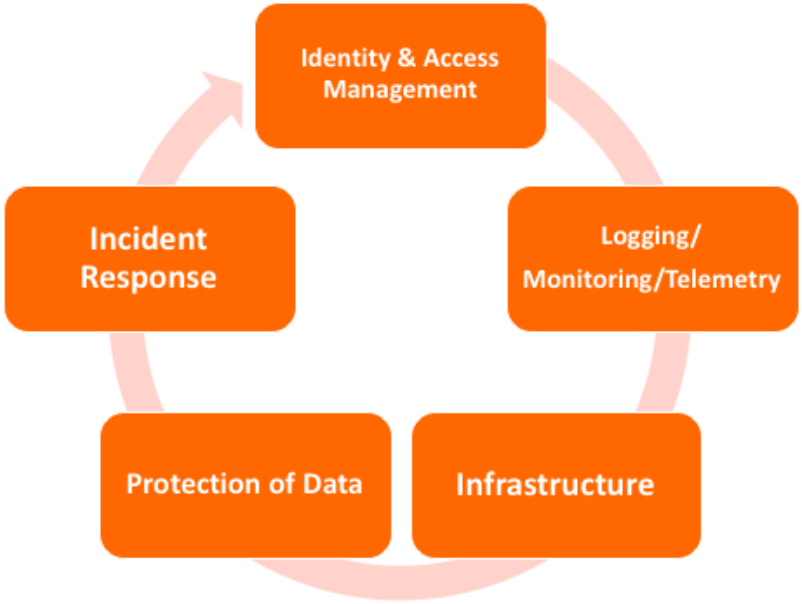
- Security is codified and applied as logic
- Security policy and strategy is designed to be preventative, not reactive
- The target architecture drives security in the pipeline
- Developers are empowered with fast feedback and automated logic which prevents mistakes from degrading the security posture, and informs them of the nature of any infractions

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

CI/CD Security – Primary Priorities



After the priorities are implemented, sprints can be added to support and build further on security priorities.

TECHTOWN

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Implementing Security in a CD Pipeline

- Define the **security strategy**
- **Automate security checks**, and remediation when possible – always providing feedback to developers
- **Promote the culture** of awareness, ownership, and continuous improvement of security outcomes
- **Focus on priorities first** – formulate overarching security policies into epics which can be executed incrementally
- Task **dedicated small teams** with establishing and increasing security velocity

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

How CI/CD security practices tie to real-world functional roles in the pipeline

- **Fast feedback** to developers via automated pipeline testing promotes continuous improvement in code creation
- Architecture is designed and refactored to allow for **reusable components tied to security**: i.e. infrastructure, change rules, tests, and policy scripts
- TDD practices are used by developers to **emplace security logic before any code is created** which could be a candidate for build

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

How CI/CD Security Practices Tie to Real-World Functional Roles in the Pipeline

- **Reusable test automation** is built by dedicated sprints and dedicated teams, engineered from requirements derived from the organization's security policy
- Gated check-ins: When **a security check fails, the pipeline stops** and the priority becomes fixing the problem
- All teams focus on **keeping the build green**, as opposed to getting code released

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Use Telemetry to help identify problems

- Identify which **types of logging** and logging levels are required for each phase of the pipeline.
- **Articulate the value** of logging and its diagnostic value
- **Evaluate logging** management tools
- Interpret and **utilize log data**
- **Share** findings to access control and security

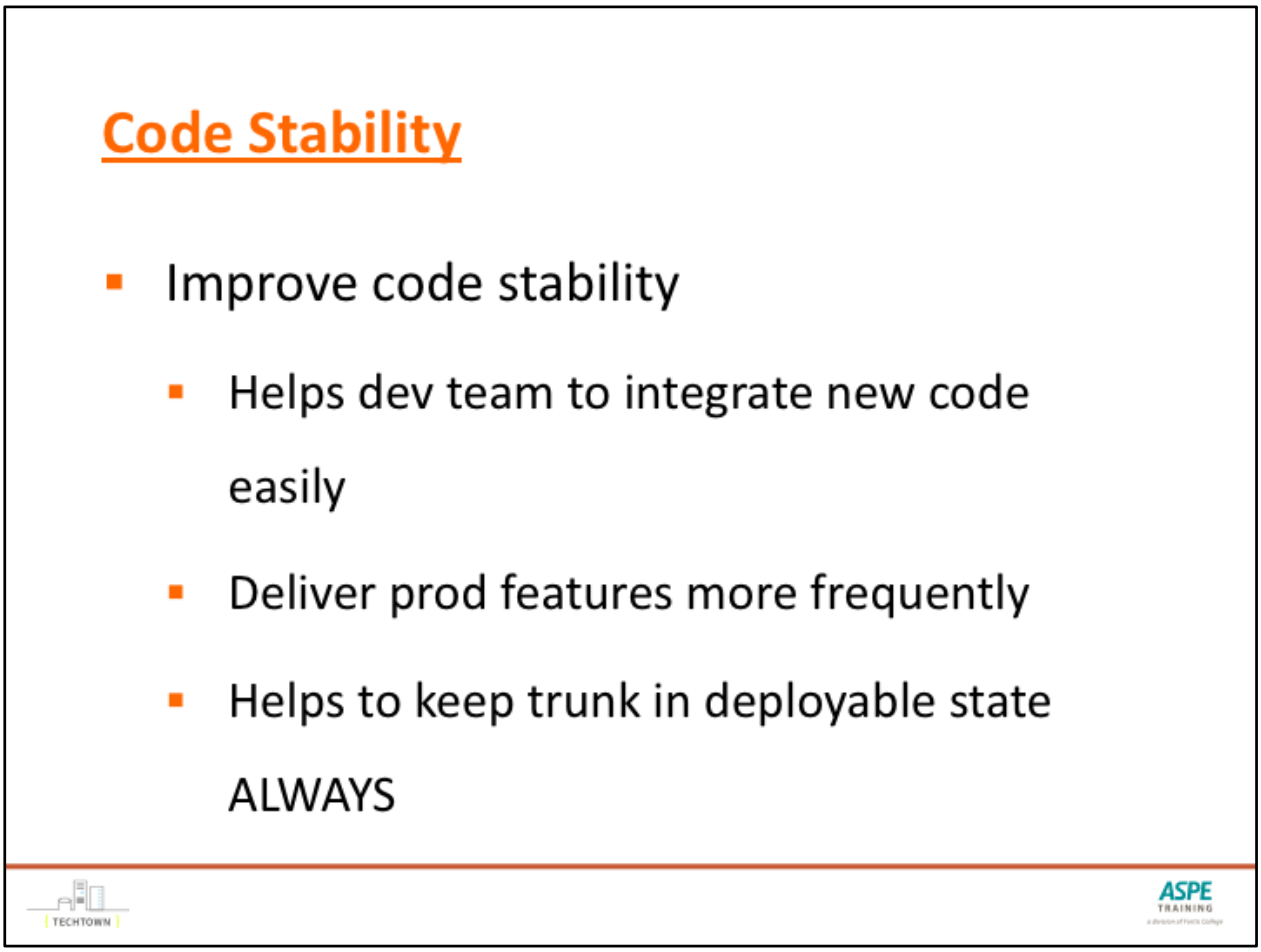
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

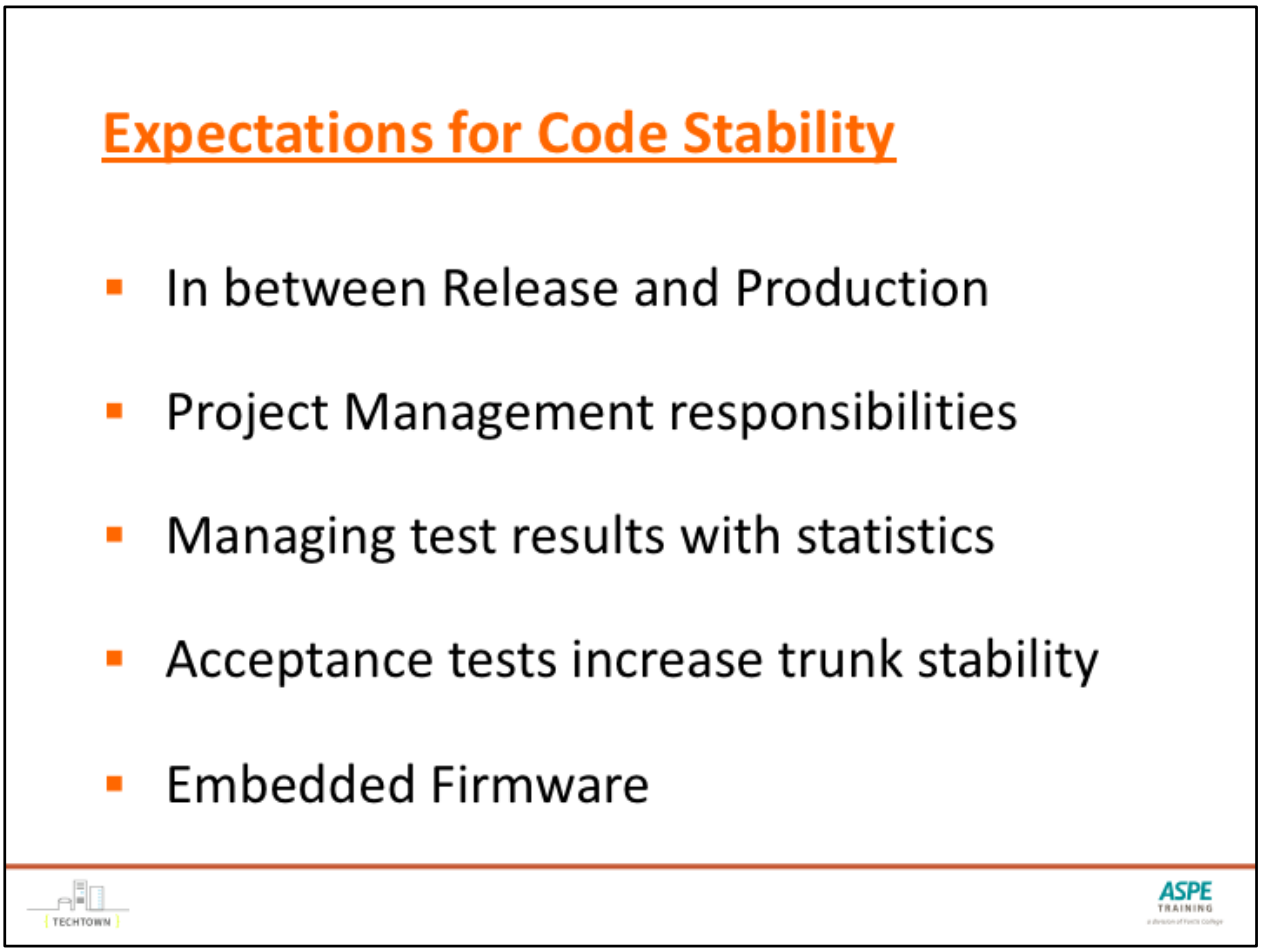
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Improving the stability of an organizations code base will make the development team more productive, because they will be able to integrate new code easily and deliver production quality features more frequently. This chapter delves into techniques which help improve stability over time.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Improving the stability of an organizations code base will make the development team more productive, because they will be able to integrate new code easily and deliver production quality features more frequently. This chapter delves into techniques which help improve stability over time.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

❖ *In between Release and Production*

- **Enabling Frequent Releases**
  - Ensure high quality to customers
  - Track defects sooner
  - Increase test pass rates
- **Making the code base releasable**
  - Reducing time & effort between Release and Production
  - Having specific code requirements on one list
  - Release Criteria for Code test rates

ASPE TRAINING

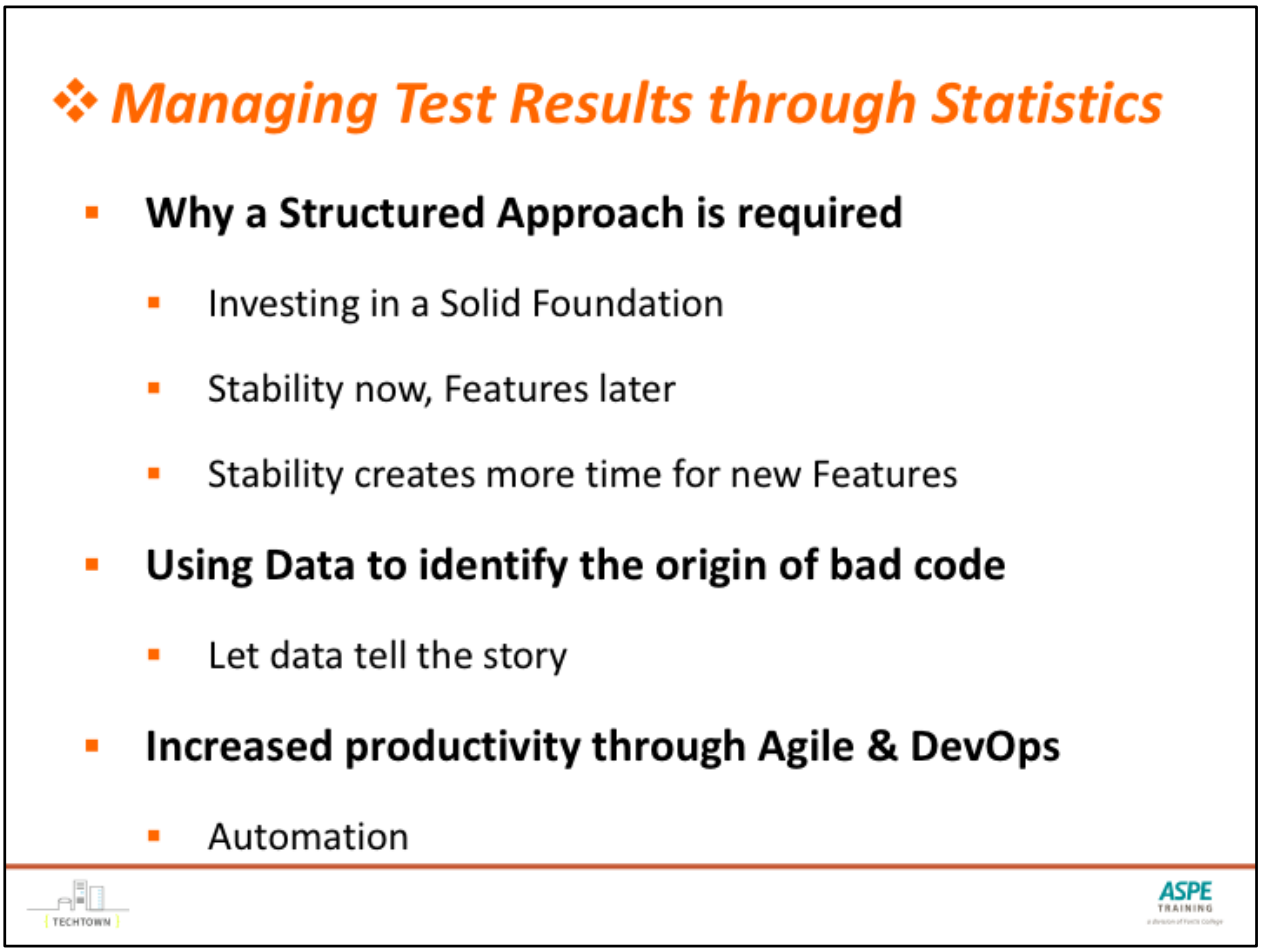
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Release Criteria from Project Management perspective
Form management, release criteria should include all of the stories need signed off, all defects fixed, and the pipeline needs to be delivering green builds with the latest code.

Stability over Features
Having a stable codebase will provide a bigger ROI than adding new features.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

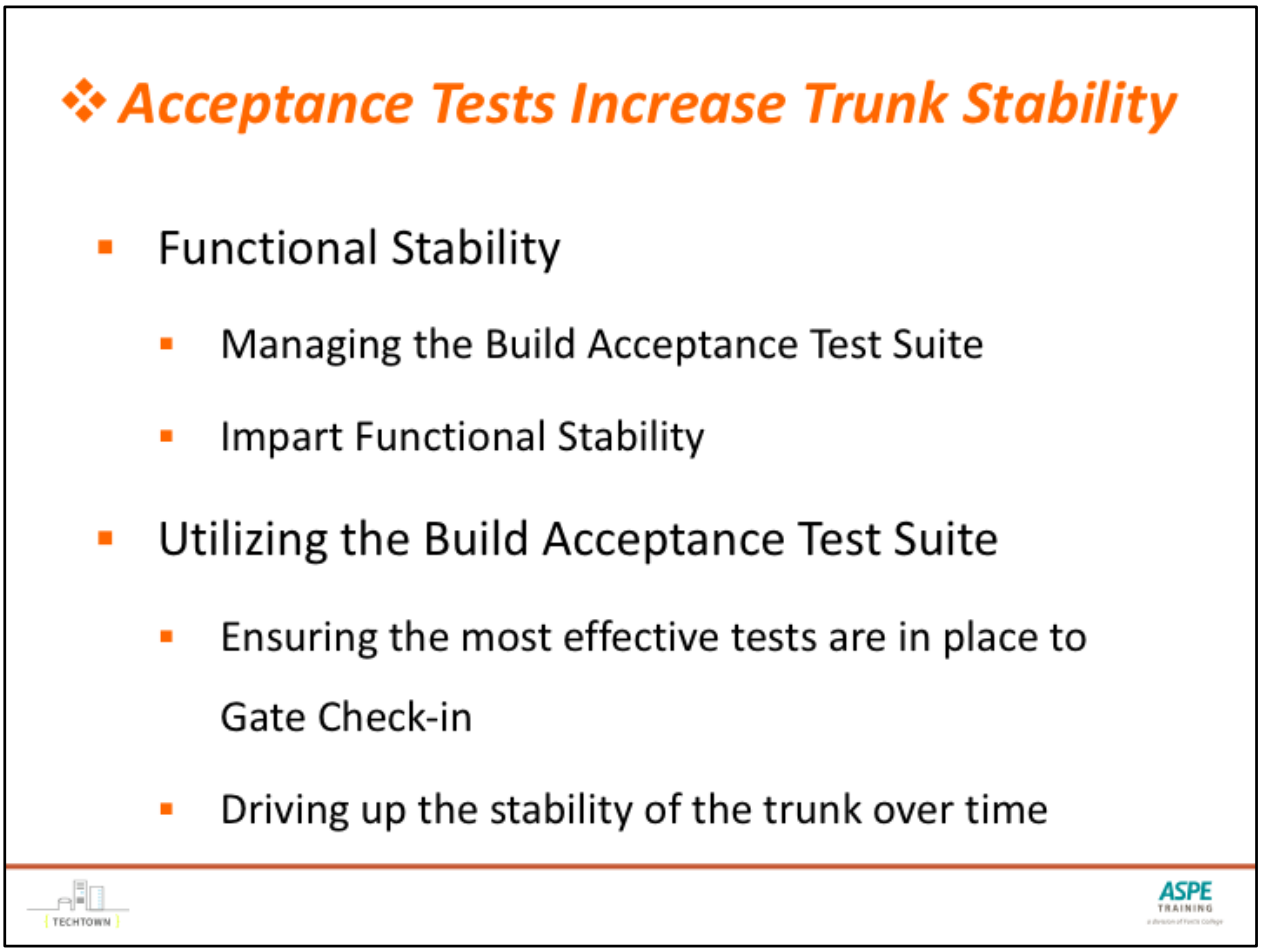


Why a Structured Approach is required
Large organizations will not achieve the full benefits of Agile development without a structured a plan for automated testing & deployment integration.

Using Data to identify the origin of bad code
Until automated testing can localize the origin of bad code, too much time will be spent determining which team submitted the offending code.

Increased productivity through Agile & DevOps
Organizations that utilize Agile & DevOps practices experience productivity gains that are dramatically larger that traditional software delivery methods.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Dramatic Pass-rate drops
When a component test suite takes a dramatic pass-rate drop, executives need to ascertain how the code that broke that many tests made it into the central repository.

Utilizing the Build Acceptance Test Suite
Acceptance tests can help to drive up pass rates over time and drive up the stability of the trunk.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

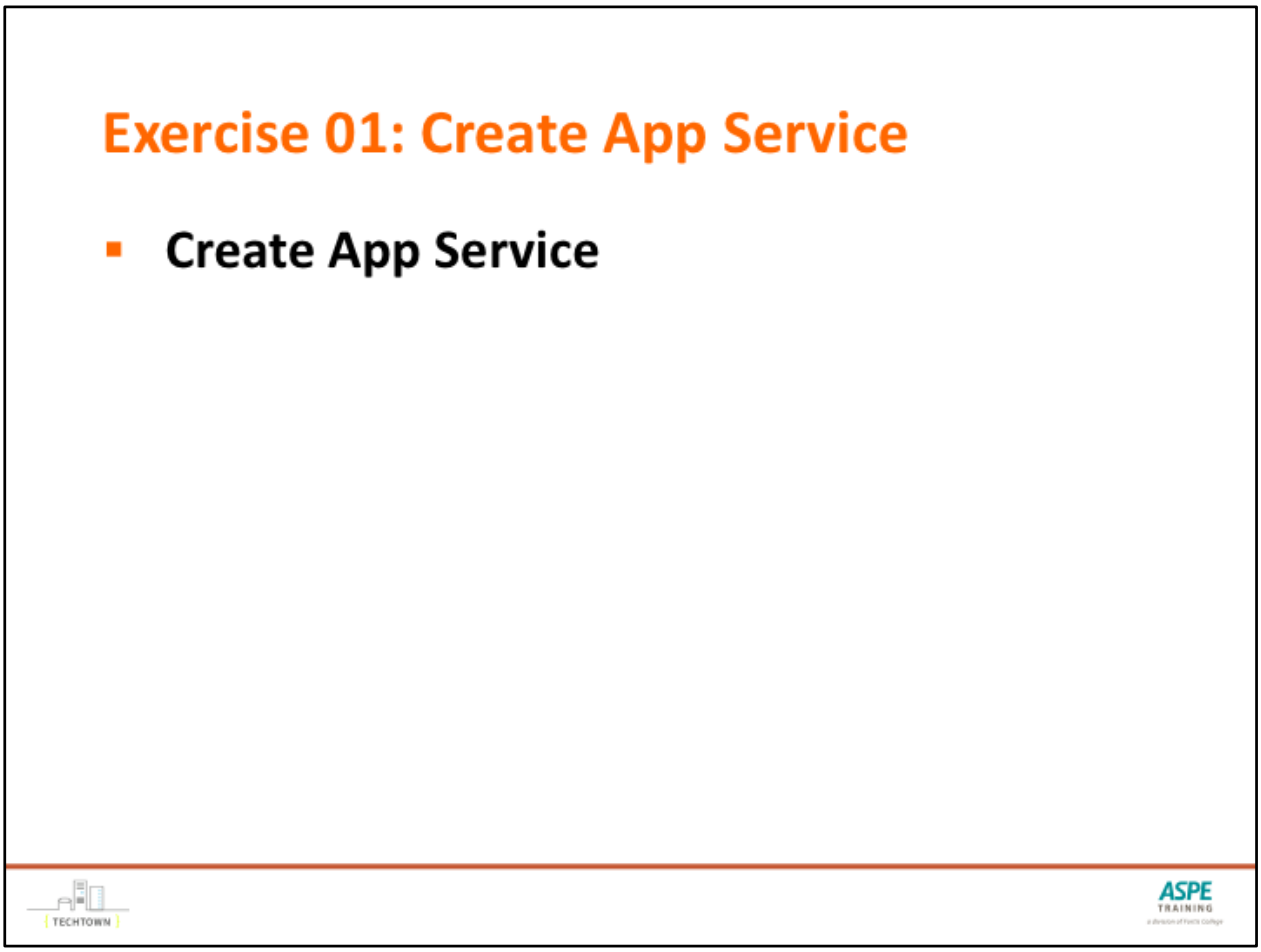
Lab: Configure Cloud Services

- Exercise 01: Create App Service
- Exercise 02: Create Azure Database

Not in Lab guide

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



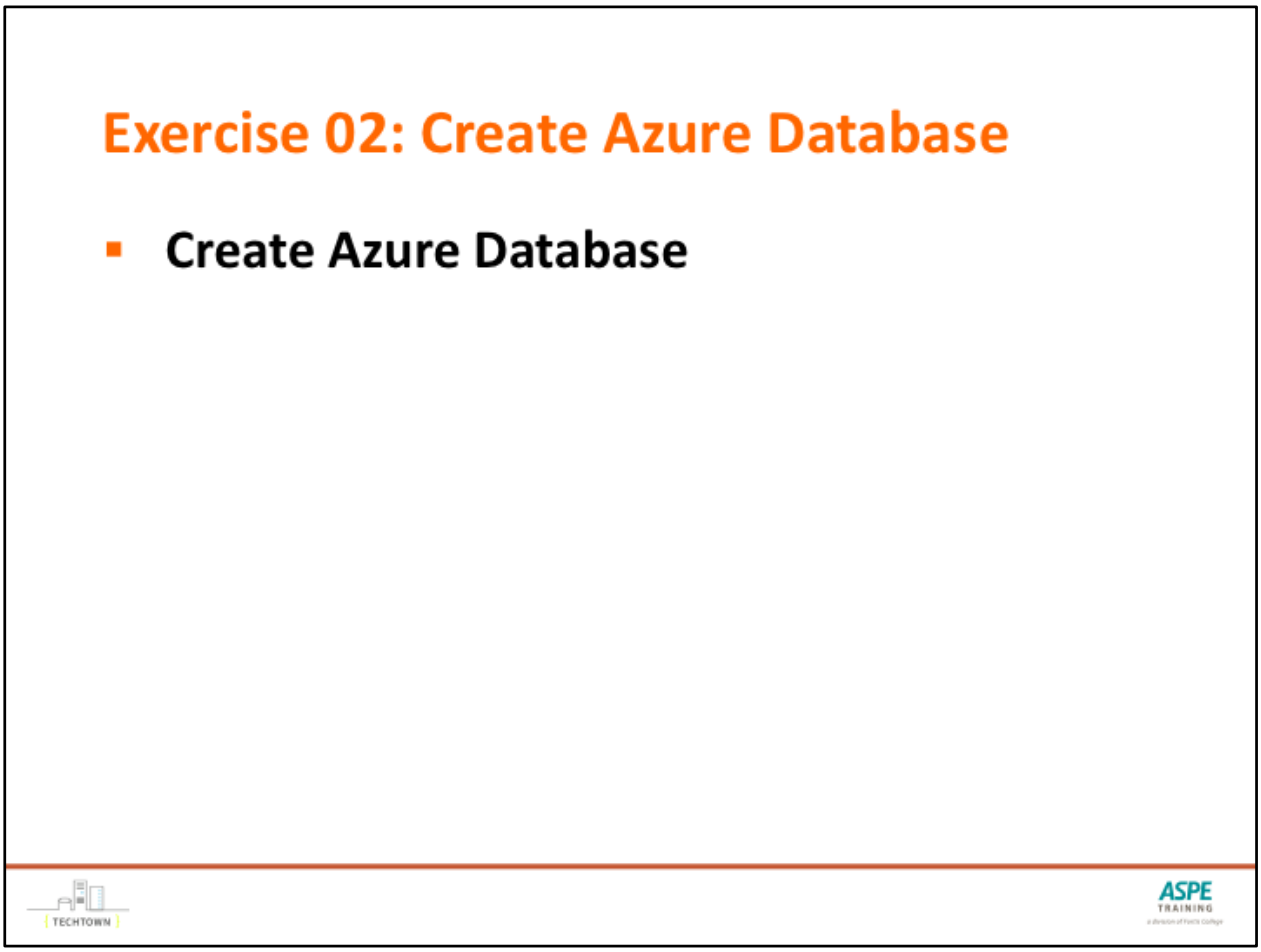
Exercise 1 Create App Service
In this exercise, we will create our application service in **Azure**

Go to azure.com -> portal
Login: DevOpsStudent@Outlook.com
Password: JustM300

Create App Services
1. Navigate to the **Azure** dashboard for your account
2. Click the **App Services** menu Item (on the left)
3. Click the **+Add** button
4. Under Web App click the Web App + SQL tile
5. Click the **Create** button (lower right)
6. Enter **ServiceReservationsXX** in the **App name** text box (Where XX is your initials)
7. Choose **Free Trial** for Subscription (or demos ?)
8. Click **Configure required settings** under **SQL Database**

Note: Do not close this window. The next exercise continues from this point.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2 Create Azure Database

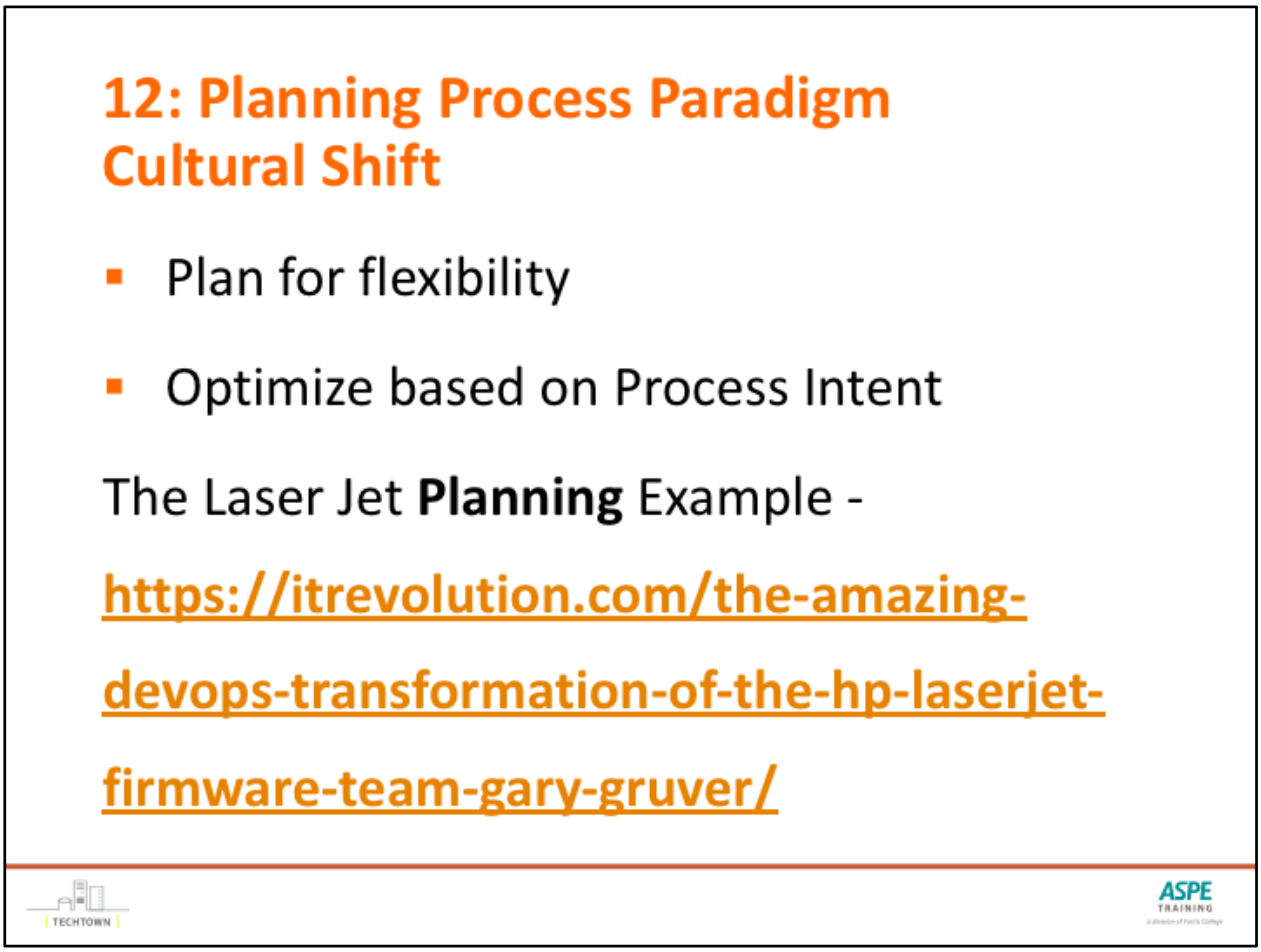
In this exercise, we will create an Azure database

1. Click the **+Create a new database** button
2. Name the database **ServiceReservationsDB**
3. Click **Configure required settings** below **Target server**
4. **Click +Create new server**
5. Enter **ServiceReservationsSQLXX** in the Server name text box (where XX is your initials)
6. Enter **SQLAdmin** in the **Server admin login** textbox
7. Enter **Pa$$w0rd** in the **Password** and **Confirm Password** textboxes
8. Select **West Central US** as the **Location**
9. Click **Select**
10. Click **Configure required settings** under **Pricing tier**
11. Select **Free** and click **apply**
12. Click **Select**
13. Check the **Pin to Dashboard** checkbox
14. Click **Create**

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

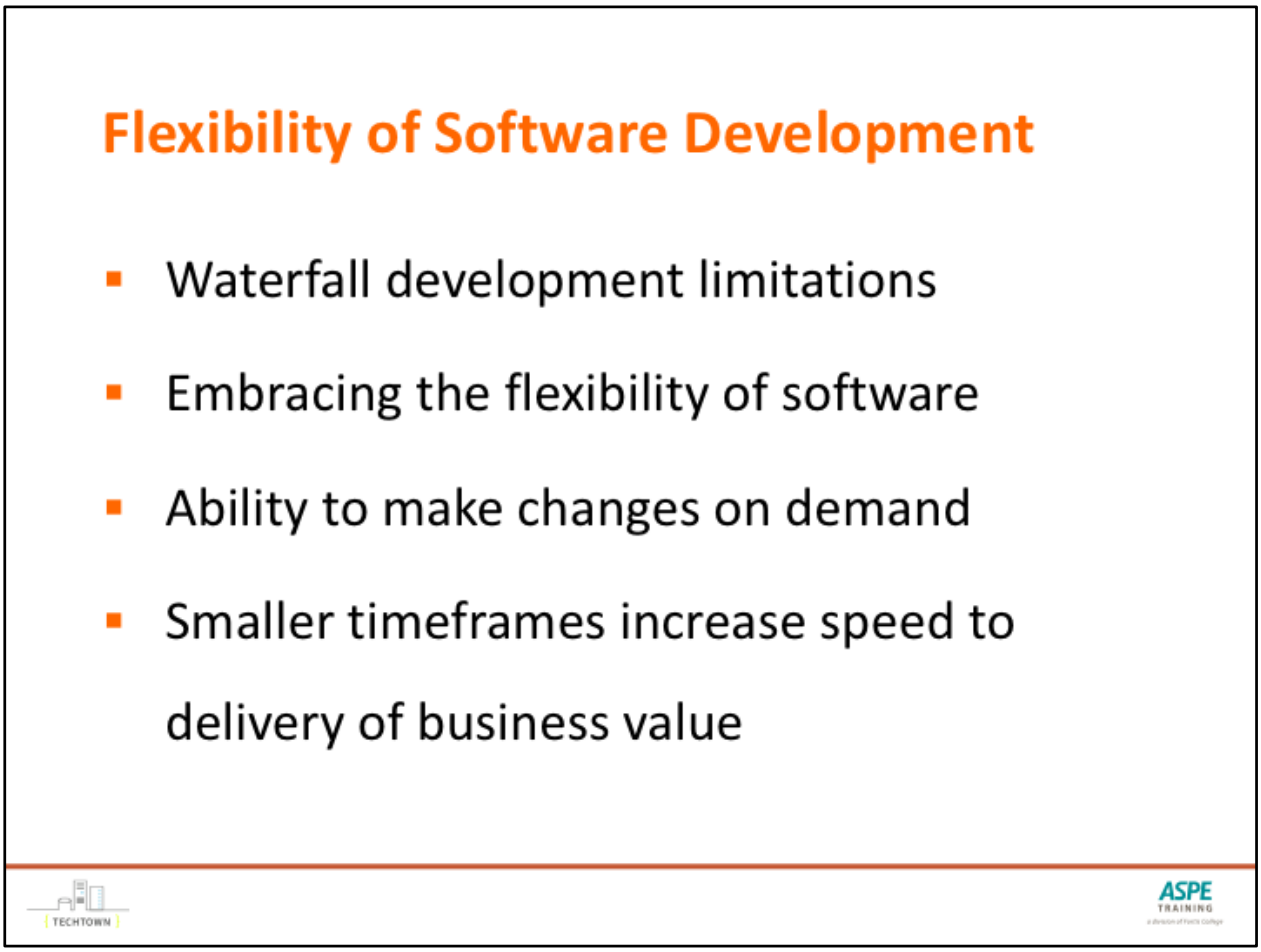
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Once sticking points, challenges, and business objectives have been identified, organizations will need to organize key processes which facilitate continuous software delivery. This chapter explores the cultural transformation that will be necessary to meet client expectations in an era where high quality and efficiency are the new norm.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Waterfall development limitations

Leadership in large organizations are usually unfamiliar with the unique characteristics of software development, especially when it comes to the application of DevOps & Agile. In the current competitive business climate, new functionality can be added at the end of the development cycle, which is not possible with Waterfall.

- Eliminates flexibility of software
- Rigidly defined feature sets
- Drives cost up higher than it should be

Embracing the flexibility of software

At the management level of a large corporation, it seems intuitive to approach software development with a traditional methodology. Software development is more flexible than most products, so it's important for executives to understand the differences that need to be applied for effective iteration planning.

- Easy to change at a reasonable cost
- Ability to make changes on demand
- Can be upgraded when technology changes

Ability to make changes on demand

If software is managed and developed correctly, it is less expensive, and easier to update than the majority of other products. If a customer requests it, or new factors necessitate it, generally speaking software can usually be changed without excessive cost, time, or energy.

- Features can be added late in the development process
- Can be upgraded at low Cost of Capacity
- Improved accuracy of planning process

Smaller timeframes increase speed to delivery of business value

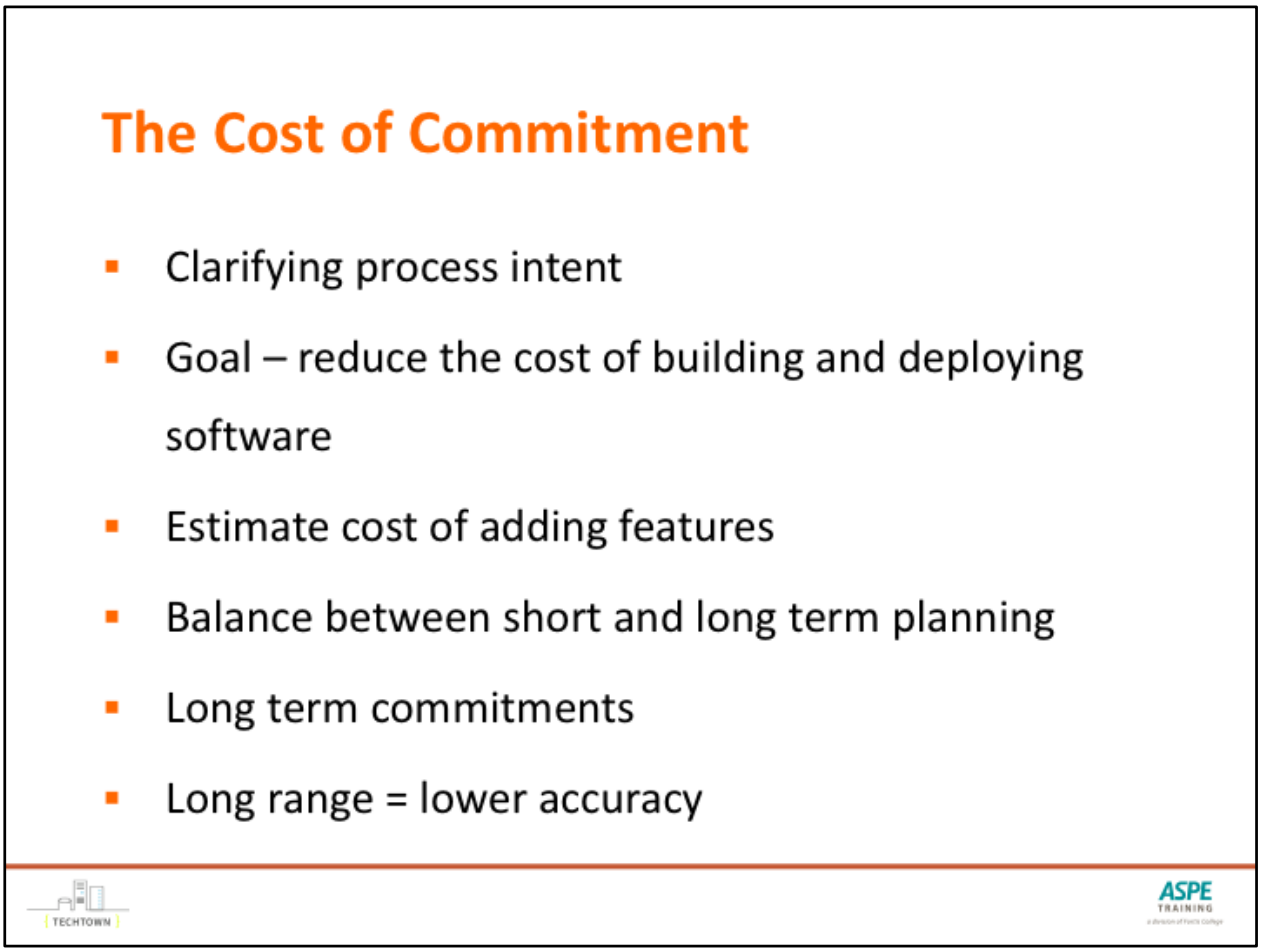
Flexibility is one of the biggest assets of software development. Breaking long-term goals into smaller tasks decreases planning time, frees up capacity, and eliminates waste when prioritizing requirements.

❖ Process Intent

- The Cost of Commitment
- Development Capacity
- Planning Requirements

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Calculating the cost

Once business objectives have been set and planning has begun, one of the main goals should be to reduce the cost of building and deploying software. In order to do so, you must first estimate the cost of adding features, strengthening the code base, identifying errors, and everything else that will factor into successful iterations.

Points for discussion:

- Responding to change
- Inevitable discovery of errors
- Built in Capacity

The cost long term

Traditional production methods employ long-term planning, because it works for most products. Software tends to be different because of its flexibility, so faster, short-term planning is generally more effective because it frees up capacity.

Points for discussion:

- Long term planning reduces flexibility
- How expensive will changes be?
- Avoid useless features

Forecasting

Effective development of software requires accurate estimates, but it is notoriously easy to make inaccurate predictions. Large organizations should estimate for objectives that are prioritized and broken down from the complex interaction of factors which impact successful delivery.

Points for discussion:

- Obtain info to support decisions
- Handling adjustments

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Development Capacity

- Capacity is a Finite Resource
- Determining required Capacity
- Planning for built in Capacity

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Determining Development Capacity

- Using Metrics to calculate Capacity
- Avoid assumptions, rely on data
- Business Value over Accuracy

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Planning Requirements

- Viewing Priorities across teams
- Avoiding team sub-optimization
- Business Priorities always come first
- System thinking

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Planning Requirements

- Prioritizing business initiatives
- Prepare for inevitable errors
- Use data to support Business Decisions

NOTES:

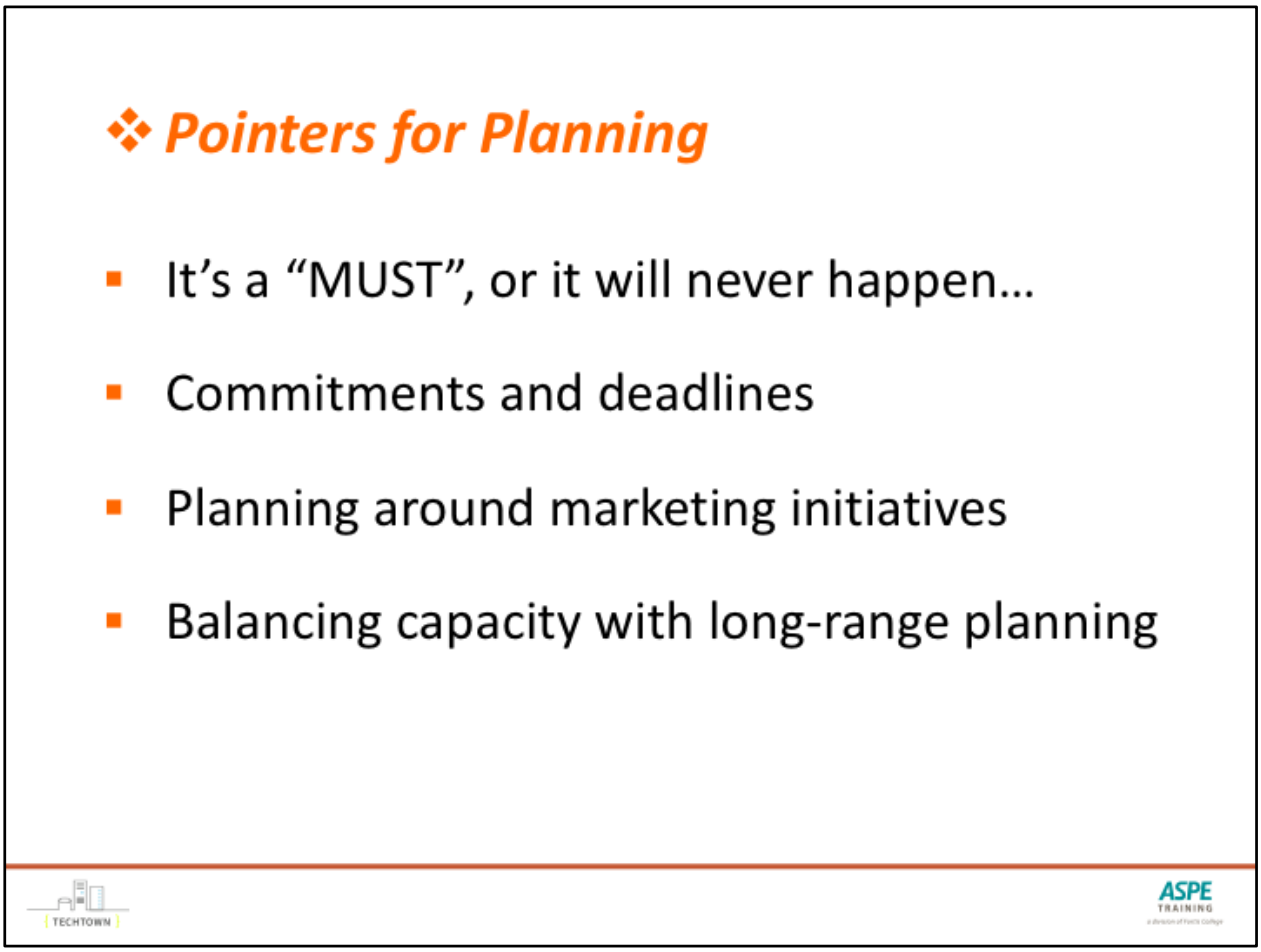
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Accuracy of Long-Range Planning

- High range accuracy is rare
- Lot of assumptions
- Discoveries on the way
- Plan for short term & build for long term

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



It's a "MUST", or it will never happen…
Many departments in larger companies have learned that if a feature is not defined as a "Must", it would never be implemented. Consequently, features that are not necessary are often defined as essential, when they are not.

Commitments and deadlines
Because of the "Must" factor, software teams often waste capacity in follow-up meetings proving to clients that "Must" features are not essential. This adversely affects deadlines, and reduces ability to make firm commitments.

Planning around marketing initiatives
Organization-wide cultural shifts are essential for development teams to continually improve and provide customer value. Developers well need to get involved in working with leadership, estimate capacity with them, and help them understand the technical factors that determine the priority of features and objectives.

Balancing capacity with long-range planning

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

“MUST”, OR it will never happen…

- All features can NOT be a “MUST”
- Estimate and plan around capacity
- Take shorter iterations

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Re-architected the code to single base
Supporting multiple branches and code sets requires more team capacity. Developing on Trunk with a single base reduces implementation resources freeing up team capacity.
So new long range commitments only will require less of the team's capacity

Separated commitment requirements

Developed a lightweight planning process

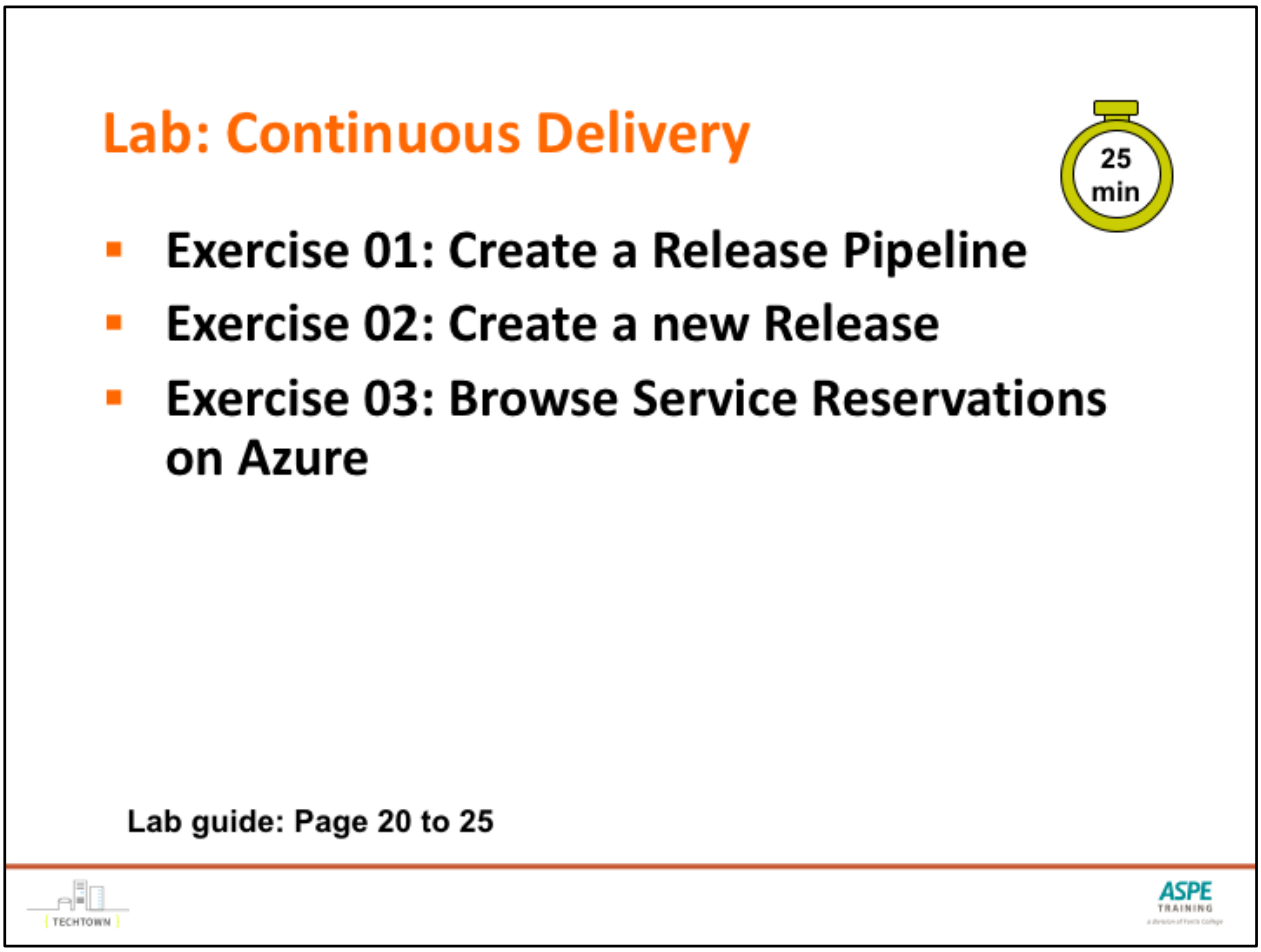
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Balancing Capacity with Long-Range Planning

- Higher capacity for developing of new features
- Minimized the Requirements inventory
- Broke the initiatives into detailed user stories
- Avoid locking in Capacity
- Keep long-range commitments to 50%
- Delivered a Prioritized Backlog

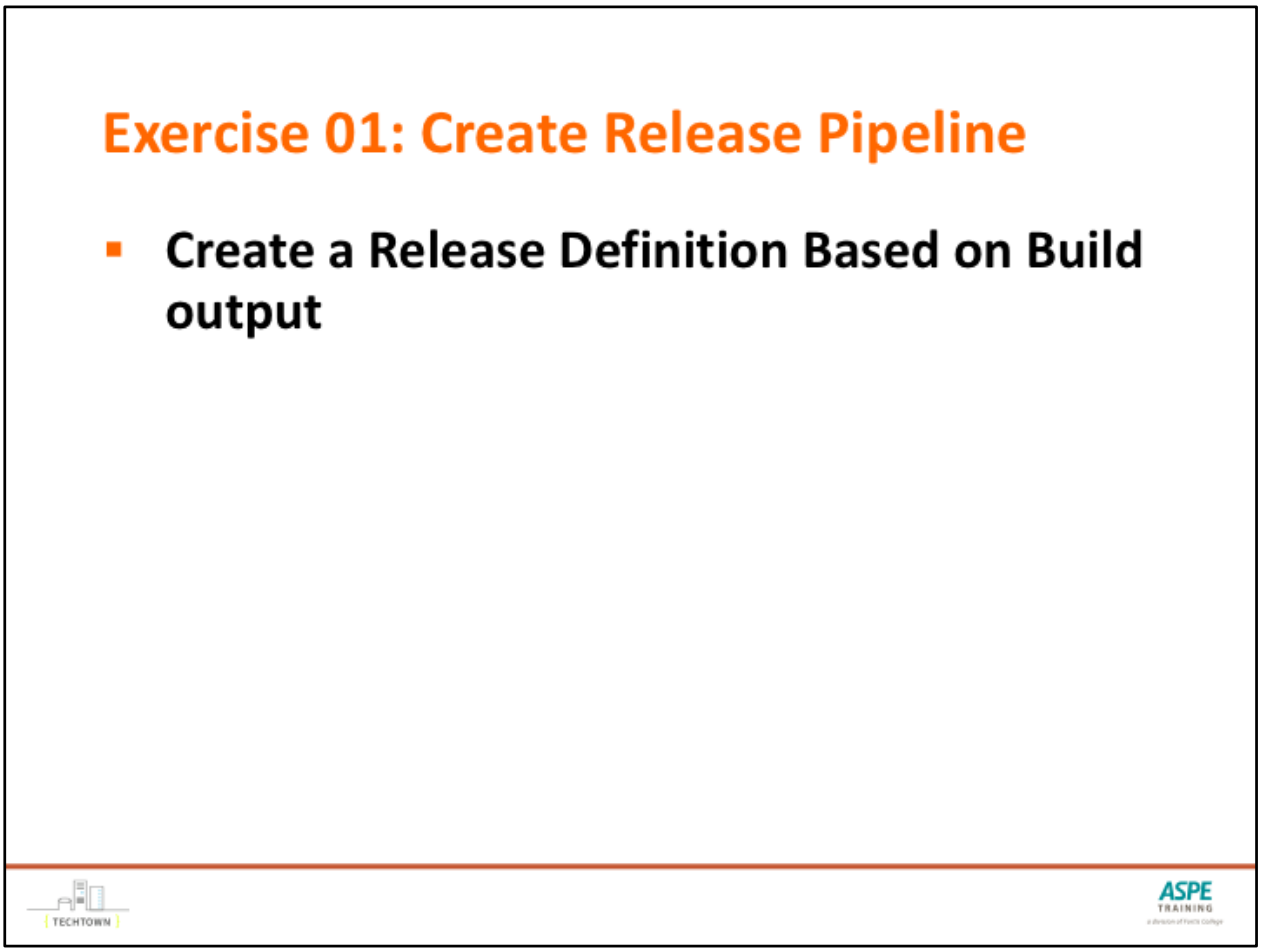
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



In this lab we will create a Release definition and deploy our solution to Azure then use a browser to review our deployed solution.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 1: Create Release Definition

In this exercise, we will create a release definition in our project based on the build output that is generated

1. In **Team Foundation Server,** navigate to **Build and Release > Releases**
2. Click the **+Create Release Definition** button
3. In the **Select a Template** window choose **Azure App Service Deployment,** and click the **Apply** button.
4. Enter "**Staging"** into the **Environment name** text box.
5. Click the text **New Release Definition** to rename the definition **Release to Staging**
6. Under **Artifact** click **Add Artifact**
7. Click **Source (Build definition)* > Service Reservation-ASP.NET-CI** > **Add**
8. From the **Tasks** menu select **Staging**.
9. Click on the **Manage** link next to **Azure subscription.**
10. Click +**New Service Endpoint** and select **Azure Resource Manager**.
11. Enter "**Service Reservations Azure App Service"** as the **Connection name.**
12. Select your **Azure** subscription from the dropdown and click **OK**.
13. If prompted enter you **Azure** credentials
14. Back in your **Release Definition** Select your **Azure** subscription from the dropdown and click **Authorize**
15. In the **App type** field, choose **Web App**
16. In the **App service name** dropdown, select **ServiceReservationsXX** (Where xx is your initials)

(continued on next page)

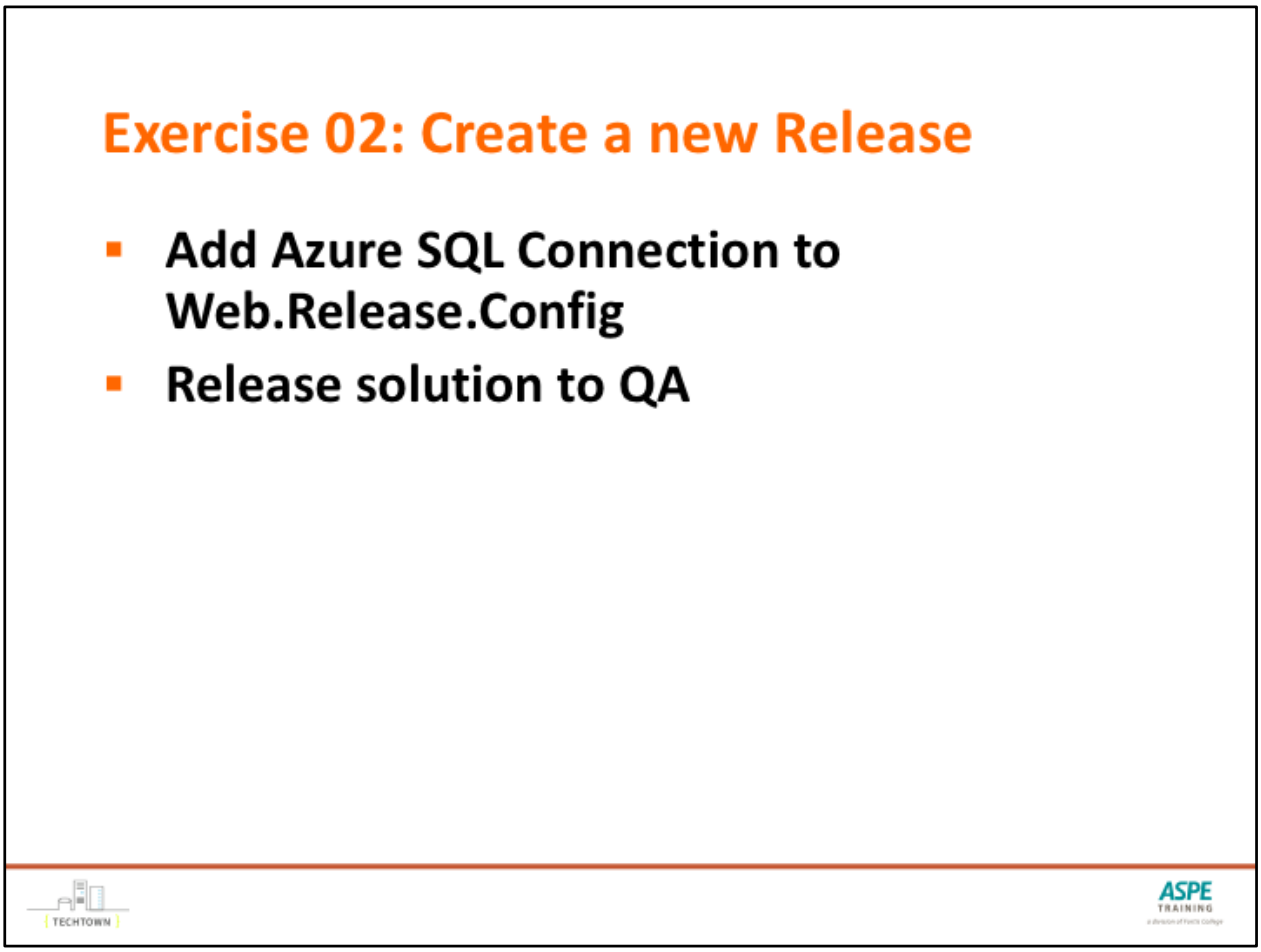
Add a Task to the Phase

1. Click the plus sign to the right of Run on agent to Add a new task to the phase
2. In the **Add task** window select the **Deploy** tab
3. Scroll down and select **Azure SQL Database Deployment** and click **Add**.
4. Drag **Execute Azure SQL: DacpacTask** above the **Deploy Azure App Service** task.
5. In the **Azure Connection Type** dropdown select Azure Resource Manager
6. In the **Azure Subscription** dropdown select your subscription.

Get the Azure Database Server Name

1. Navigate back to your **Azure** subscription dashboard
2. Under **SQL Databases** select **ServiceReservationDB (check the server column for the server called servicereservationssqlXX where xx is your initials)**
3. Copy the **Server Name**
4. Switch back to the **Release Definition**, and paste the server name that you just copied into the **Azure SQL Server Name** field
5. Enter **ServiceReservationsDB** as the Database name.
6. Enter your username in the **Server Admin Login** field, and enter your password into the **Password** field
7. In the type field, choose the **SQL DACPAC** File option
8. In the **DACPAC** File text box enter ****/*.dacpac**
9. Accept the remaining defaults, click the **Save** button., and click **OK** to save your release definition.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2: Create a new Release
Now we are almost ready to attempt to perform a release, but first we need to update the Web Application configuration so that it knows where our new Azure database server is.

Add Azure SQL Connection to Web.Release.Config

1. Navigate back to **Visual Studio**, go to the **solution explorer**, and click on the **Web.Config** file
2. In the Code Window, line 12 where you see "**DefaultConnection**"
3. In **Web.Release.config** code window, we need to uncomment the **connectionString** in line 11.
4. Change the name of it from "**MyDB**" to "**DefaultConnection**",
5. Change the name of the **connectionString** to what we have in **Azure,** which is:
6. To find it, navigate back to Azure, click **SQL databases** from the menu items, and choose **ServiceReservationsDB**
7. Click the **Show database connection strings** option and copy the connection string by clicking on the icon next to the field where it resides.
8. Paste the connection string into the **connectionString** property of the **DefaultConnection** in **Web.Release.Config**

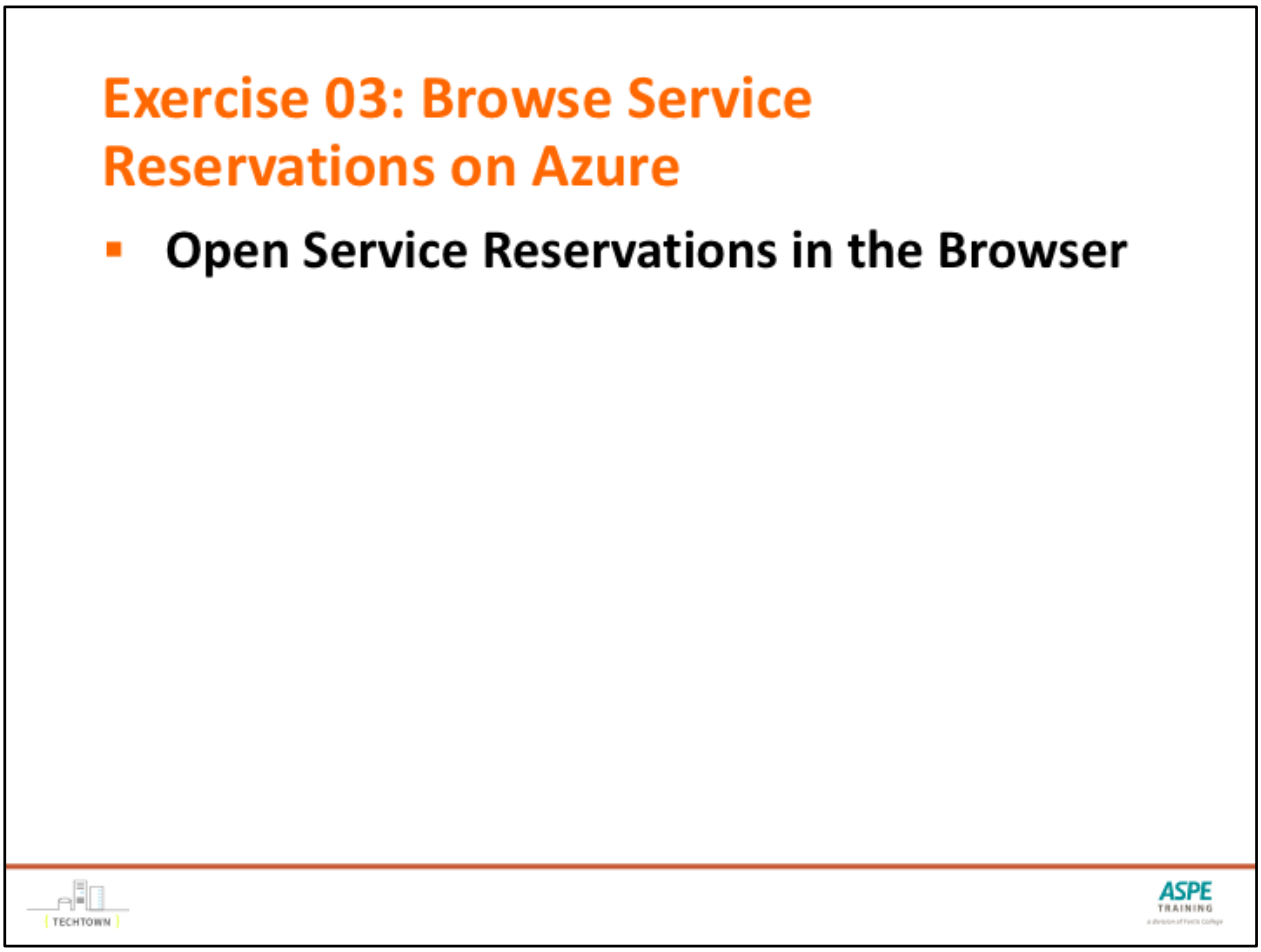
NOTE – this takes place in the Web **Release.config tab in **Visual Studio***

(continued on next page)

Release solution to Staging

1. On the **Azure DevOps** site navigate to **Pipelines > Release**
2. On Release Pipeline Page click **New > Release**
3. In the Create new release dialogue, accept the defaults, and click the **Create** button.
4. Monitor your release by clicking the Release link in the Release XXXXX has been created field that is highlighted in green.
5. Here, we can take a look to ensure that everything is going according to plan.
6. Notice, beneath **Deployment Status**, it reads **IN PROGRESS** highlighted in blue.
7. Click on the **Logs** tab, and we can watch the release run. Notice that the blue icon next to **Initialize Agent** turns green.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 3: Browse Service Reservations on Azure

In this exercise, we will open the Service Reservation project, and register with the necessary information

Open Service Reservations in the Browser

1. Open the browser and navigate to **ServiceReservationsXX.AzureWebsites.net**
2. Click Register
3. Enter email
4. Enter password
5. Click register

   Now you should be logged in

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Determining the first steps for the implementation of Agile principles in an organization can be daunting. In this chapter, we will look at how companies can prioritize business goals and create clear objectives that adhere to their unique requirements and client expectations.

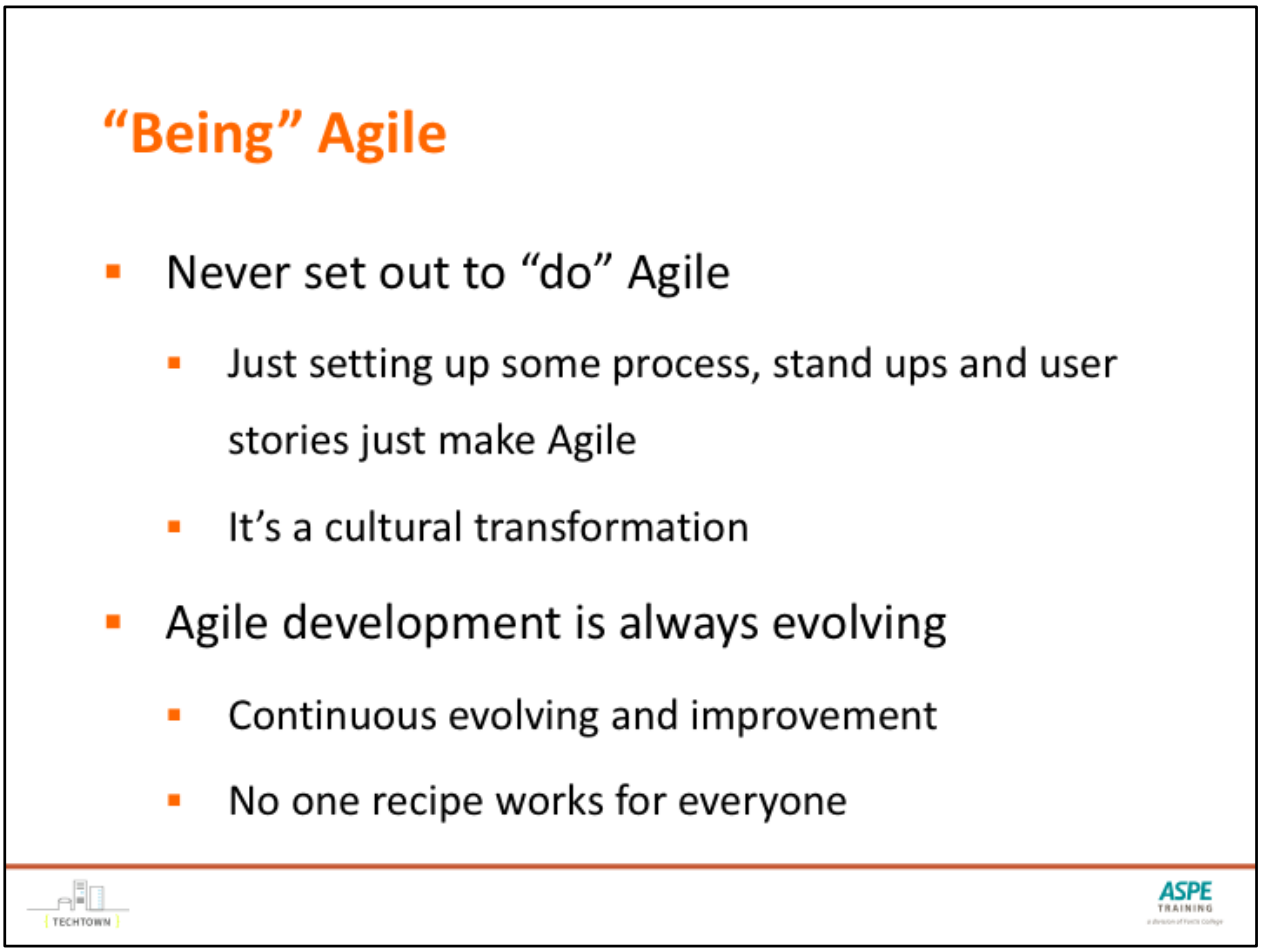
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Don't "Do" Agile
Many companies make the mistaking of trying to "do" Agile, which inherently suggests that it is something that will eventually be done. The truth is, in order to get the full benefits of this methodology, organizations will need to be, and remain Agile.

Clear Objectives
The key to implementing Agile begins with clear business goals that are shared and agreed upon from the Executive level, the development level, and the Operations level.

Collaboration
Organization-wide coordination is absolutely necessary for effective Agile & DevOps processes. Realistically, a huge amount of effort will be required from everyone who is engaged in the many facets of the endeavor.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

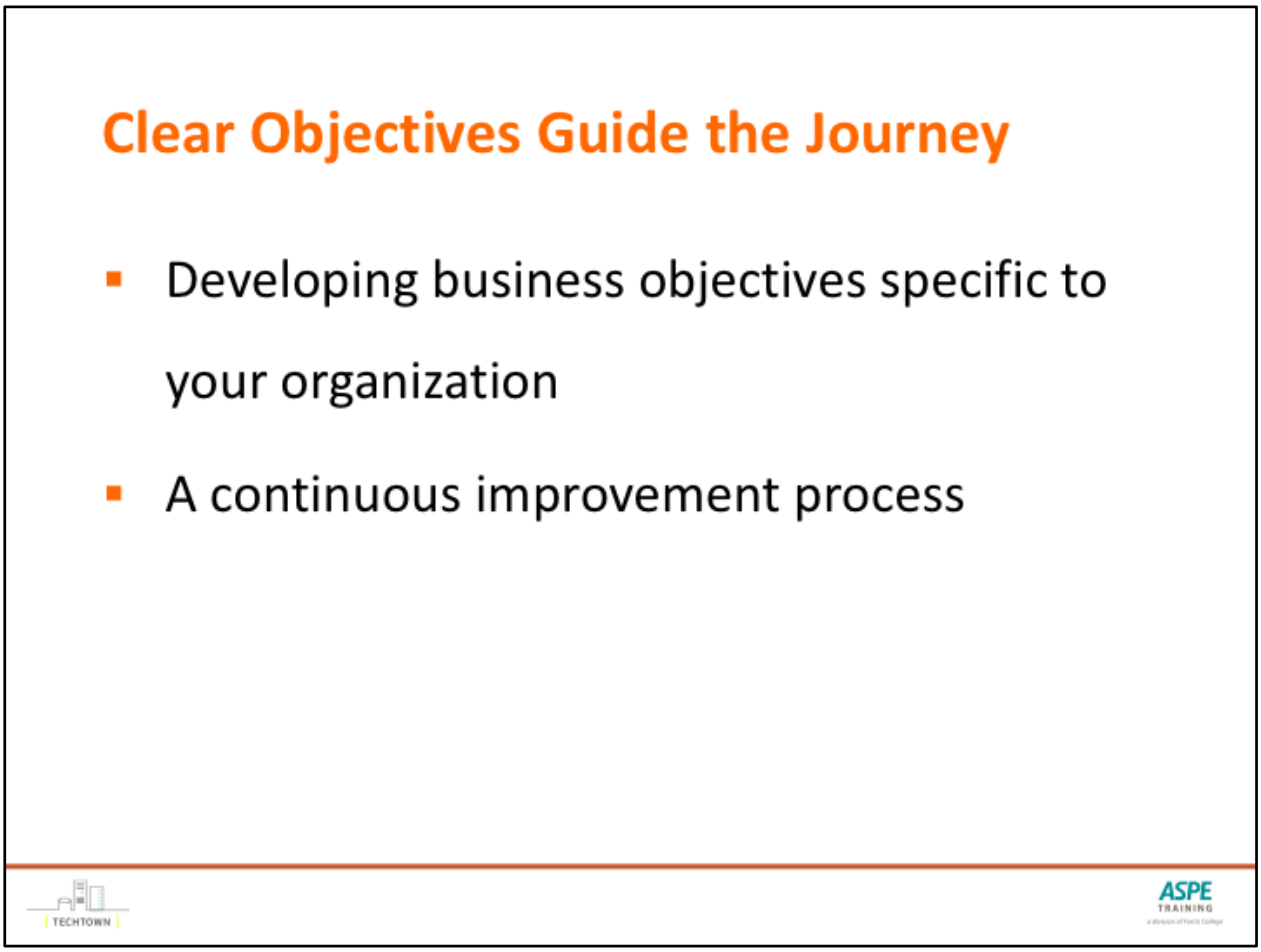


10 Objectives of an Agile Coach
http://www.keystepstosuccess.com/top-10-objectives-of-agile-coach/

Never set out to "do" Agile
When organizations set out to "do" agile, they will most likely adopt some of the processes, see some improvements, and leave it at that. Stand up meetings and User stories are not going to get you the full benefits of Agile, so it is much better for companies to look at it as a cultural transformation in the way they develop software.

Agile development is always evolving
Technology is continuously evolving, and Agile tools and processes will inevitably evolve along with it. One of the core tenants of Agile practices is Continuous Improvement, so if you are not evolving, you are not Agile.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



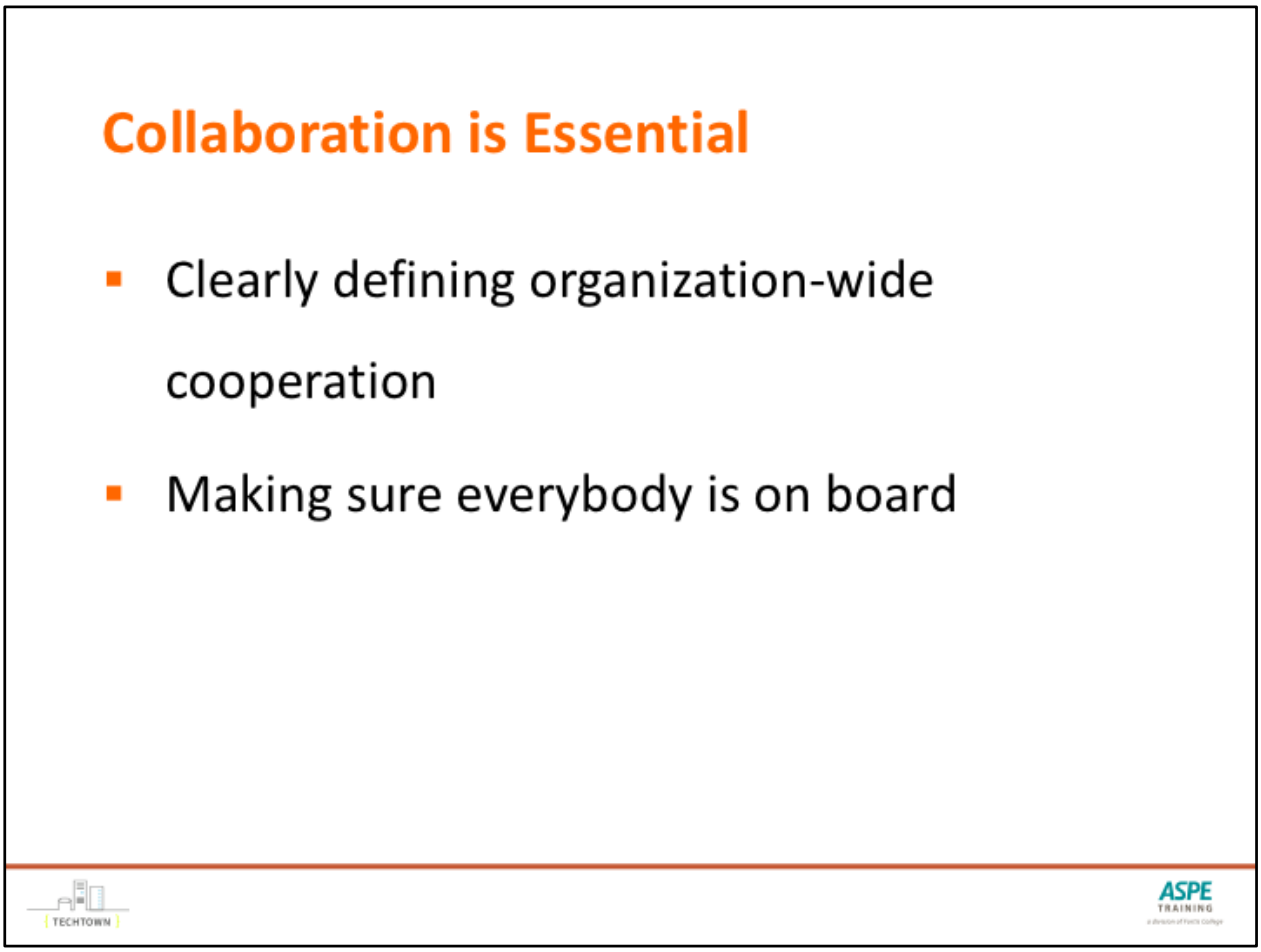
Developing business objectives specific to your organization
Writing code is similar in large and small organizations, but the processes for planning, integration, and qualifying, and releasing it are vastly different.

A continuous improvement process
In order to be Agile, companies will need to continuously improve processes, speed to market, and customer value.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

NOTES:

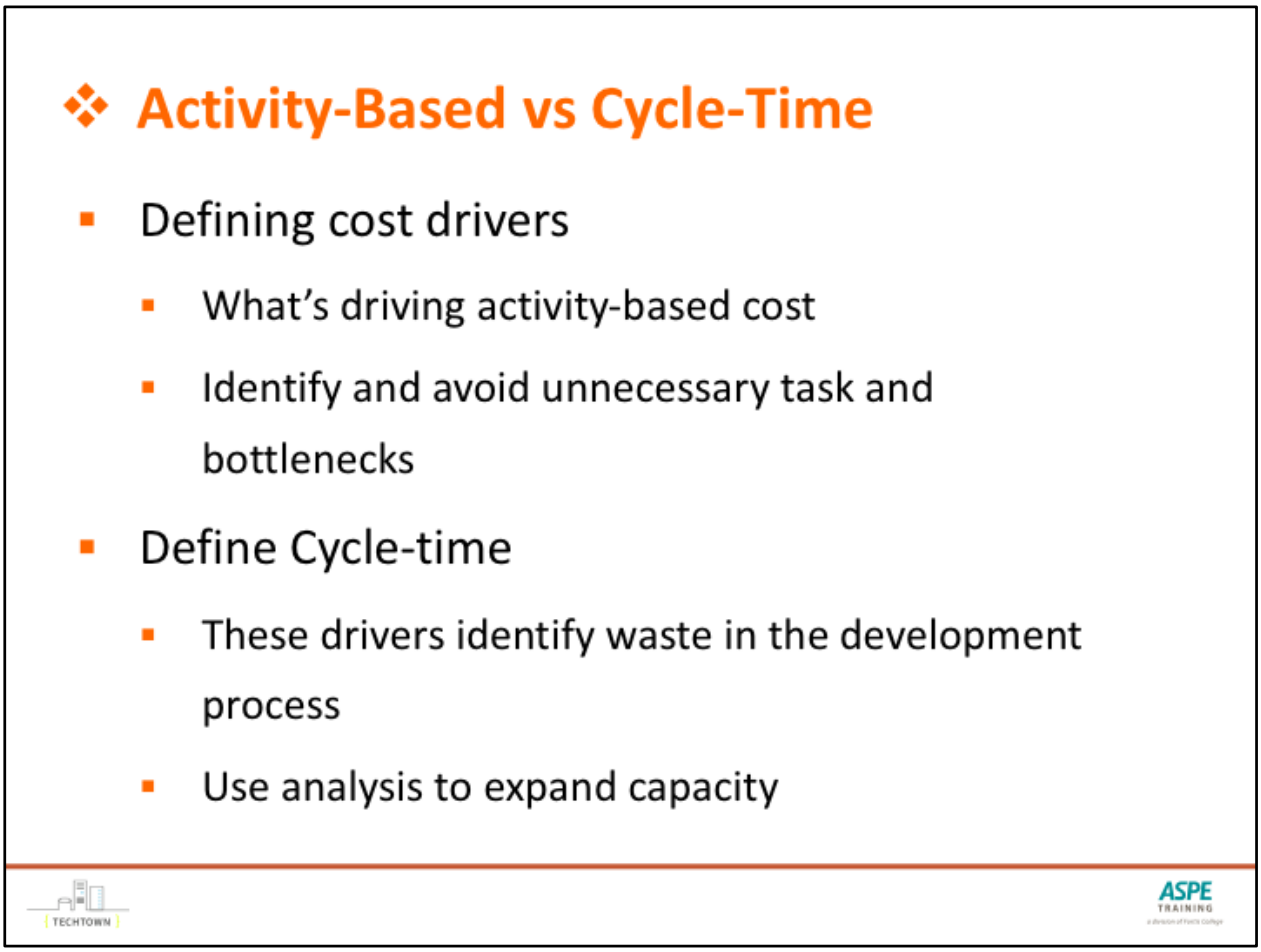
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Clearly defining organization-wide cooperation
Continuous Improvement begins with a clear understanding of how people are spending their time. Communication between teams, executives, developers, and operations must be consistent, specific, and transparent.

Making sure everybody is on board

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

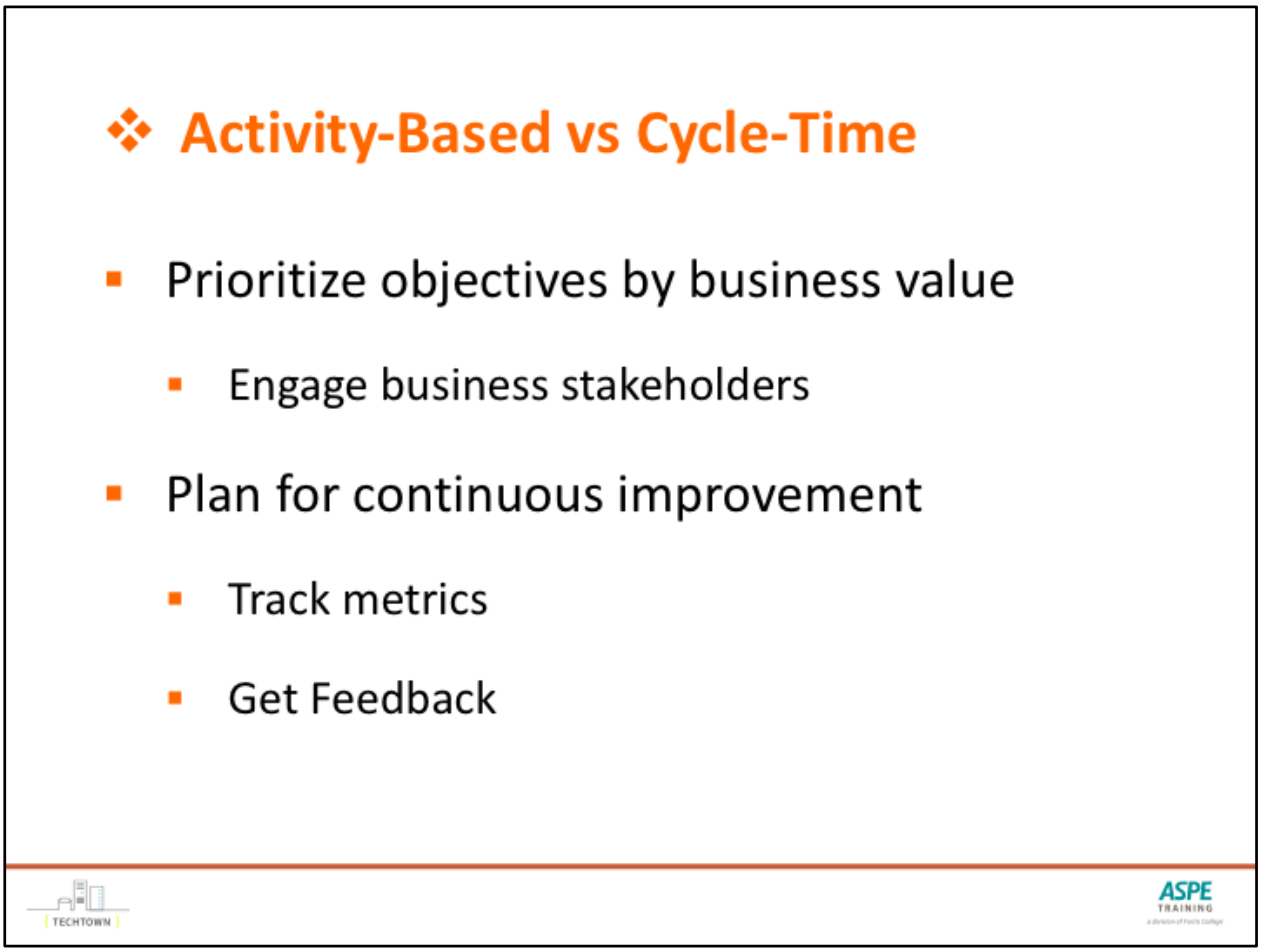


Defining cost & Cycle-time drivers
Understand what is driving activity-based cost

Prioritize objectives by business value
Determine iteration objectives that will provide the most value to the customer, and prioritize them accordingly to achieve continuous improvement.

Plan for continuous improvement
Track metrics and plan on improving processes for each iteration.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

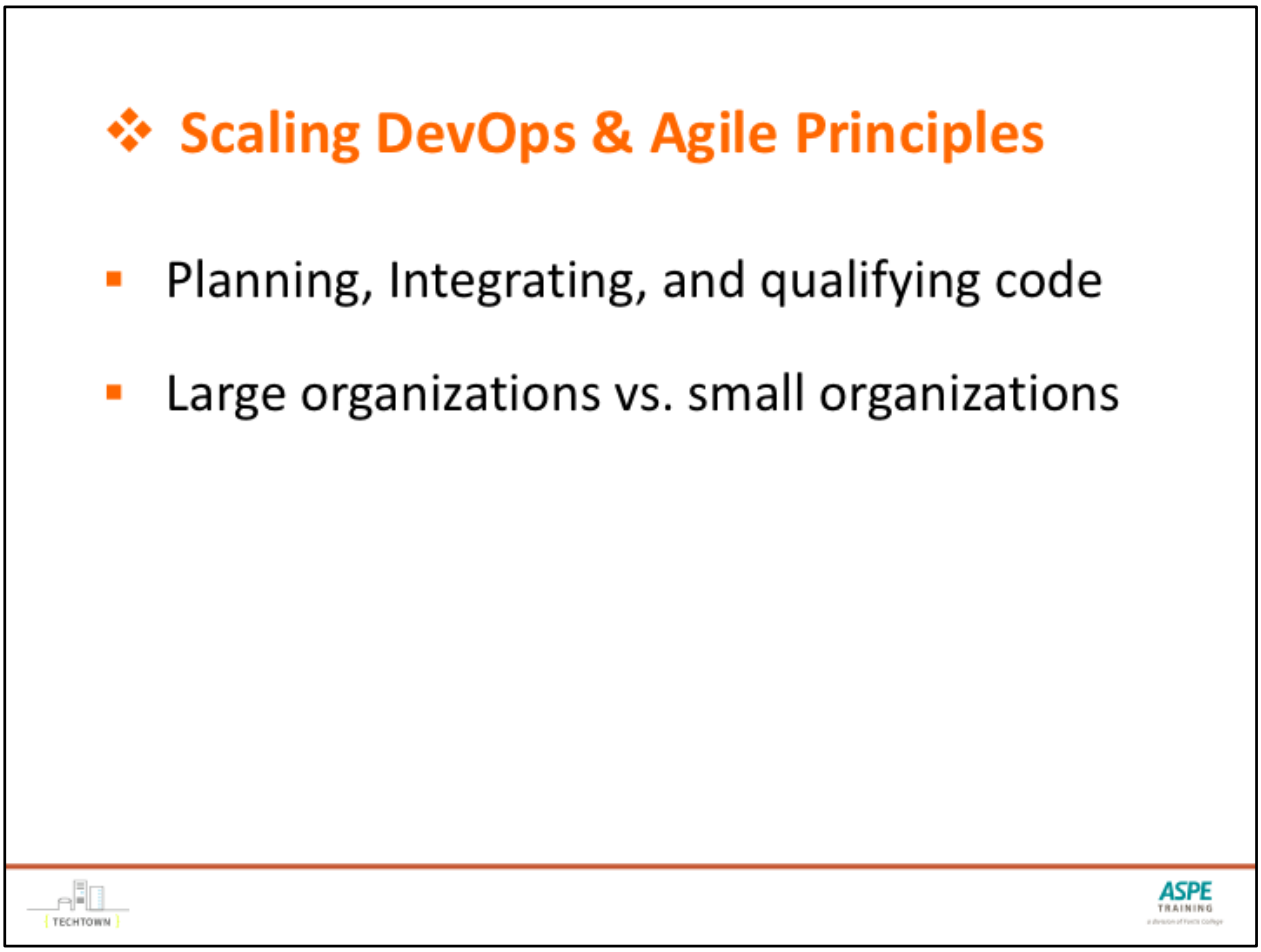


What is a Cycle-Time Driver?
Cycle-Time Drivers identify waste in development processes.

Identify unnecessary tasks, bottlenecks, and time-wasting action items
Analysis of these items is necessary to facilitate meaningful improvement.

Using analysis to expand capacity
Beginning with goals that obtain the biggest return in the smallest amount of investment will free up team capacity for more improvement.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Many organizations struggle to provide business results with Agile implementation because they focus on the "right" application methods as opposed to applying principles at scale. Large companies will need to focus on the latter, with an emphasis on customer value.

Writing code in large and small organizations is similar, but planning, integration, qualification, and delivery is directly affected by size and scope. In larger companies, the application of immediate feedback & frequent releases will take more time and more resources, and preparation for these factors should be considered at the executive level.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Planning, Integrating, and Qualifying Code

- Maintaining your code for consistent release
- Eliminate Unnecessary Planning
- Modify processes for flexible response to customer feedback

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Large Vs Small Organizations

- Writing code in large and small Orgs is similar

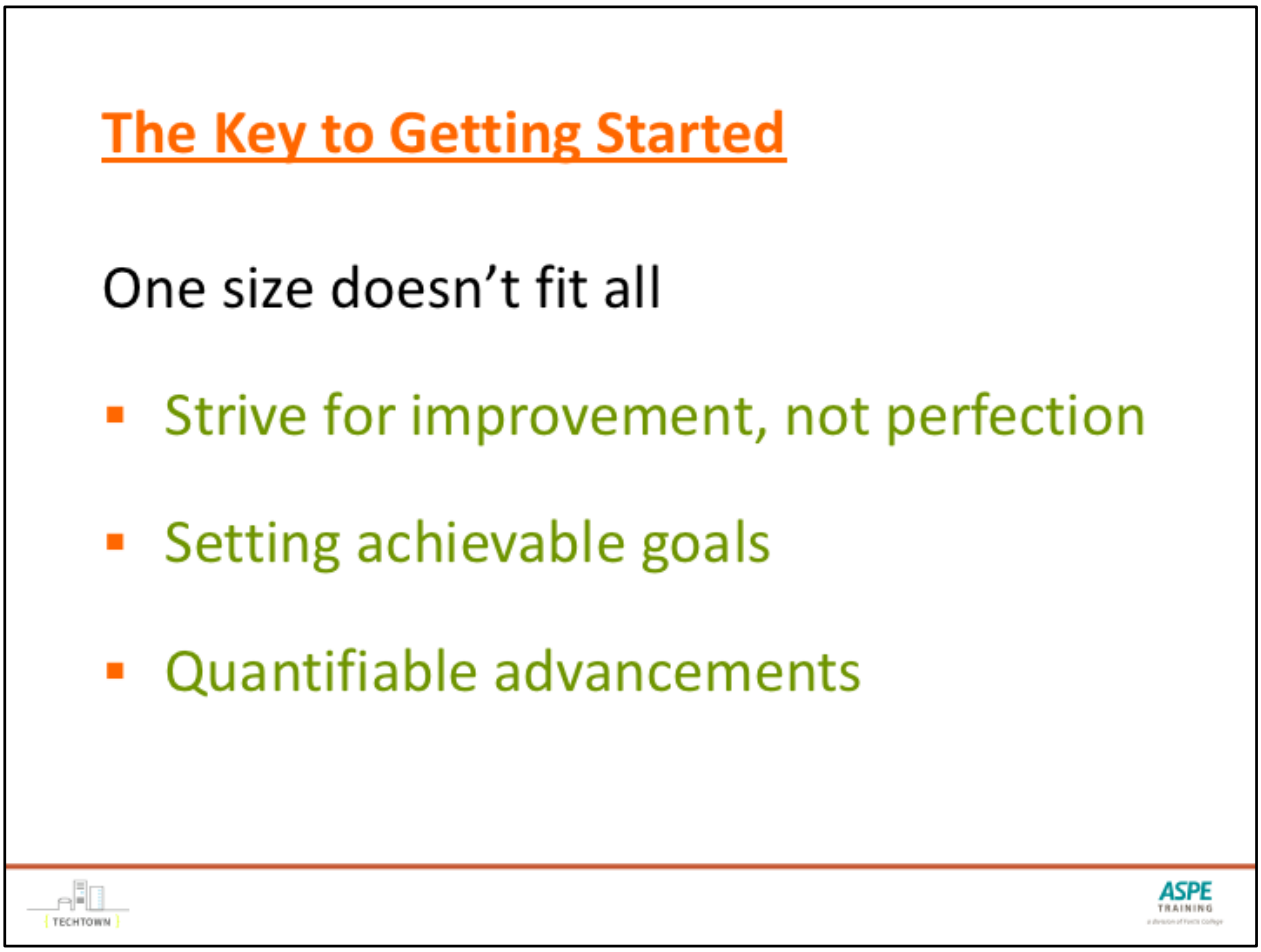
BUT

- Planning, integration, qualification, and delivery might be different
- In larger companies, the application of immediate feedback & frequent releases will take more time and more resources
  - Considered at the executive level

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



When it comes to Agile and DevOps, one size does not fit all, because different businesses have different needs, different scales, and different resources. In the nest pages, we will talk about the importance of determining a starting point and provide some recommendations on how to choose it based on your organizations unique needs.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Key Pointers to Getting Started

- Start small, focus on value
- Prioritize actions that show improvement
- Unique Considerations for your Organization
- Executives Drive Cultural and Technical Change
- Don't let the Magnitude of overall change prevent initial steps

NOTES:

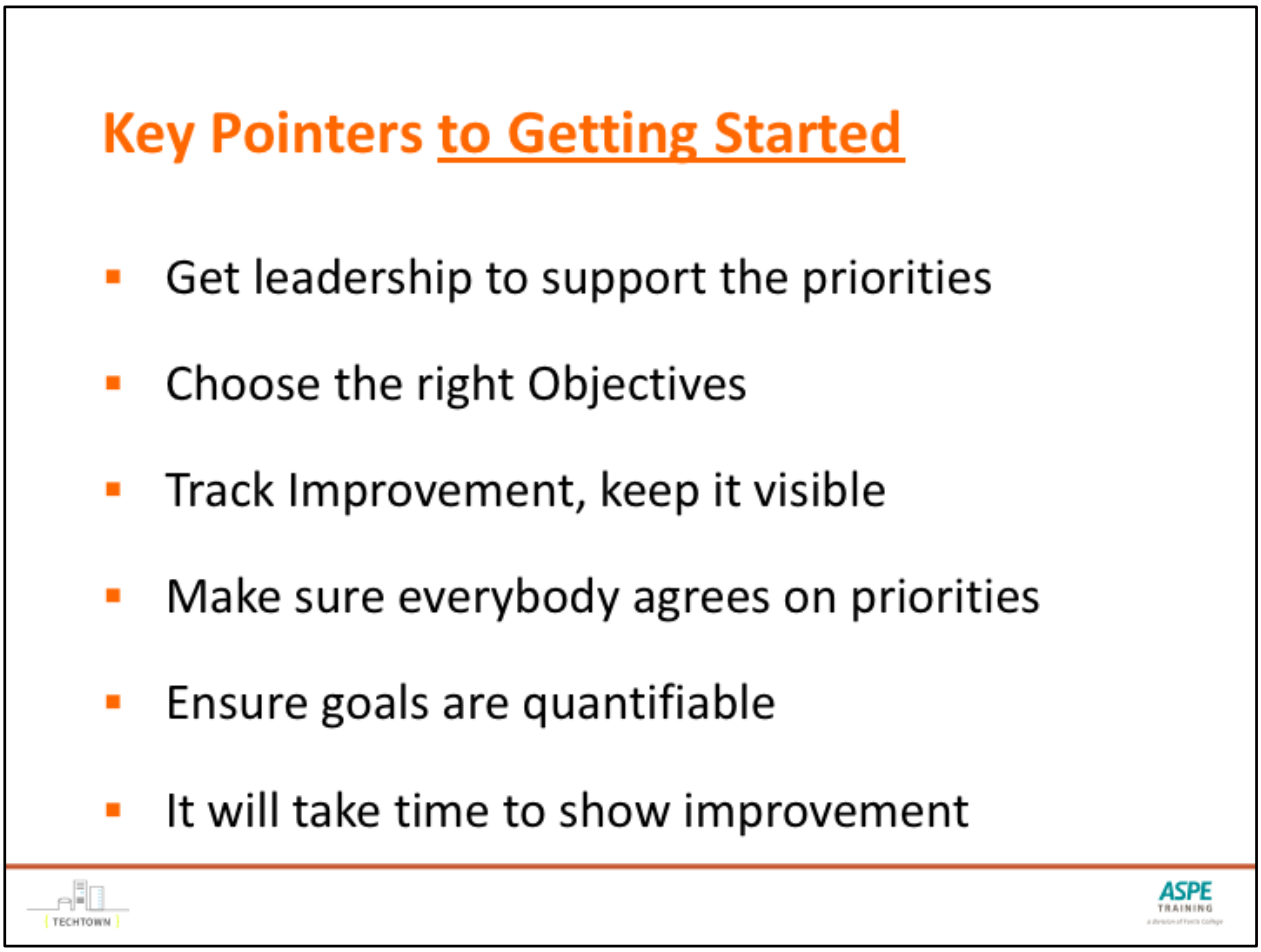
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Key Pointers to Getting Started

- Enterprise-level continuous improvement
- Begin with clear business objectives
- Choose the right team to lead the process
- Communicating the importance of Transformation
- Change is difficult, not changing is inevitable failure

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Choose the right Objectives

Executives will need to be involved in helping to prioritize objectives. Objectives should be directly related to:

Implement a cultural shift the empowers developers to take ownership of all code checked in a production like environment

Coordinating Development and Operations tools & processes

Organizationwide embrace of processes that capitalize on software flexibility

Exploiting the benefits of automation to create a stable code base & deploy defect-free features in a small time-frame

Track Improvement, keep it visible

Having a development process that integrates stable code across the enterprise takes time, but it is one of the most effective ways of aligning objectives & tools across multiple teams. Since this is a slow process, progress and challenges need to be tracked & reviewed daily, so executives have data to back up each process change, one iteration at a time.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Questions?

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Feedback Email

If you haven't already, you will receive an email similar to the one here asking for your feedback on the course.

This Evaluation will ask your feedback on many aspects of your training, including the course materials, instructor facilitation and labs where applicable.

We appreciate you completing this survey to help us continue to better our content and deliveries in order to serve you better.

TECHTOWN

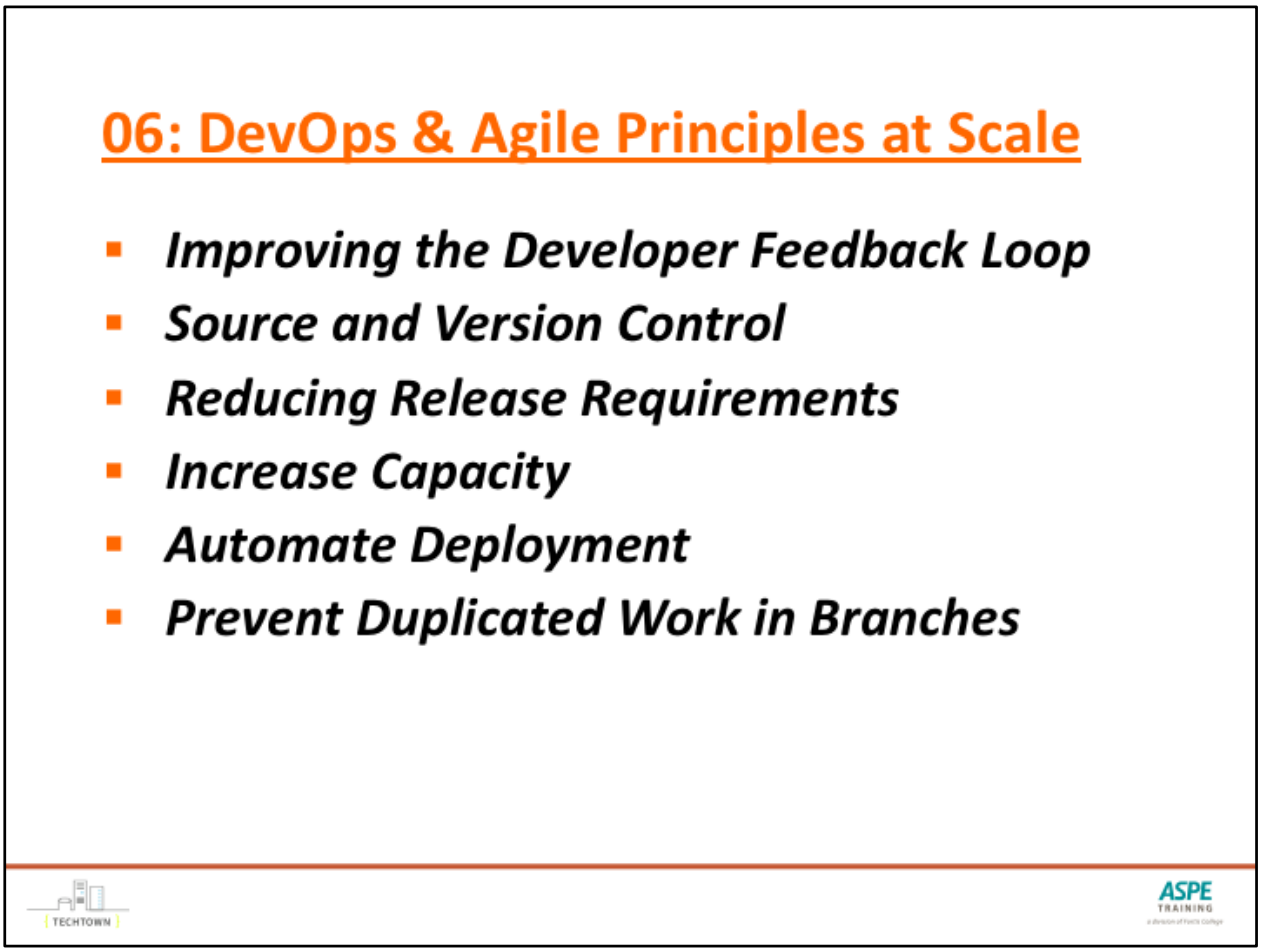
ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

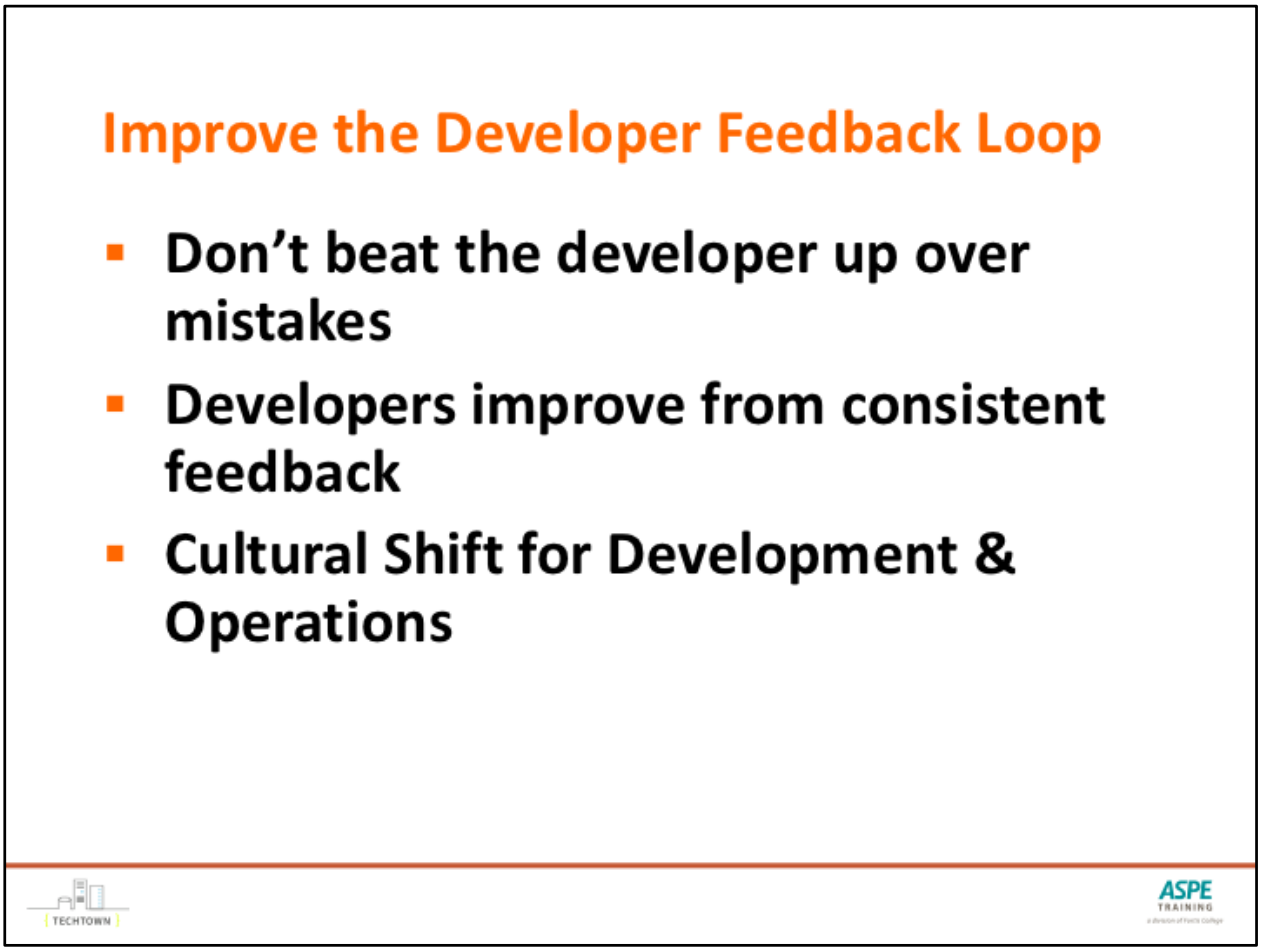
NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



One of the key elements that will help organizations compete effectively in the Agile era of software development is the ability to release small batches of functionality at a rate that is economically feasible for the company. In this chapter, we will discuss the importance of cultural application, technical processes, and how to find a balance between the two.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Don’t beat the developer up over mistakes
Developers think they have written good code until they are told otherwise. If they get in trouble for bad code that was submitted weeks or months ago, this communication will not be very productive. (“What code? When? Are you sure it was my code?”) **Prioritize:**

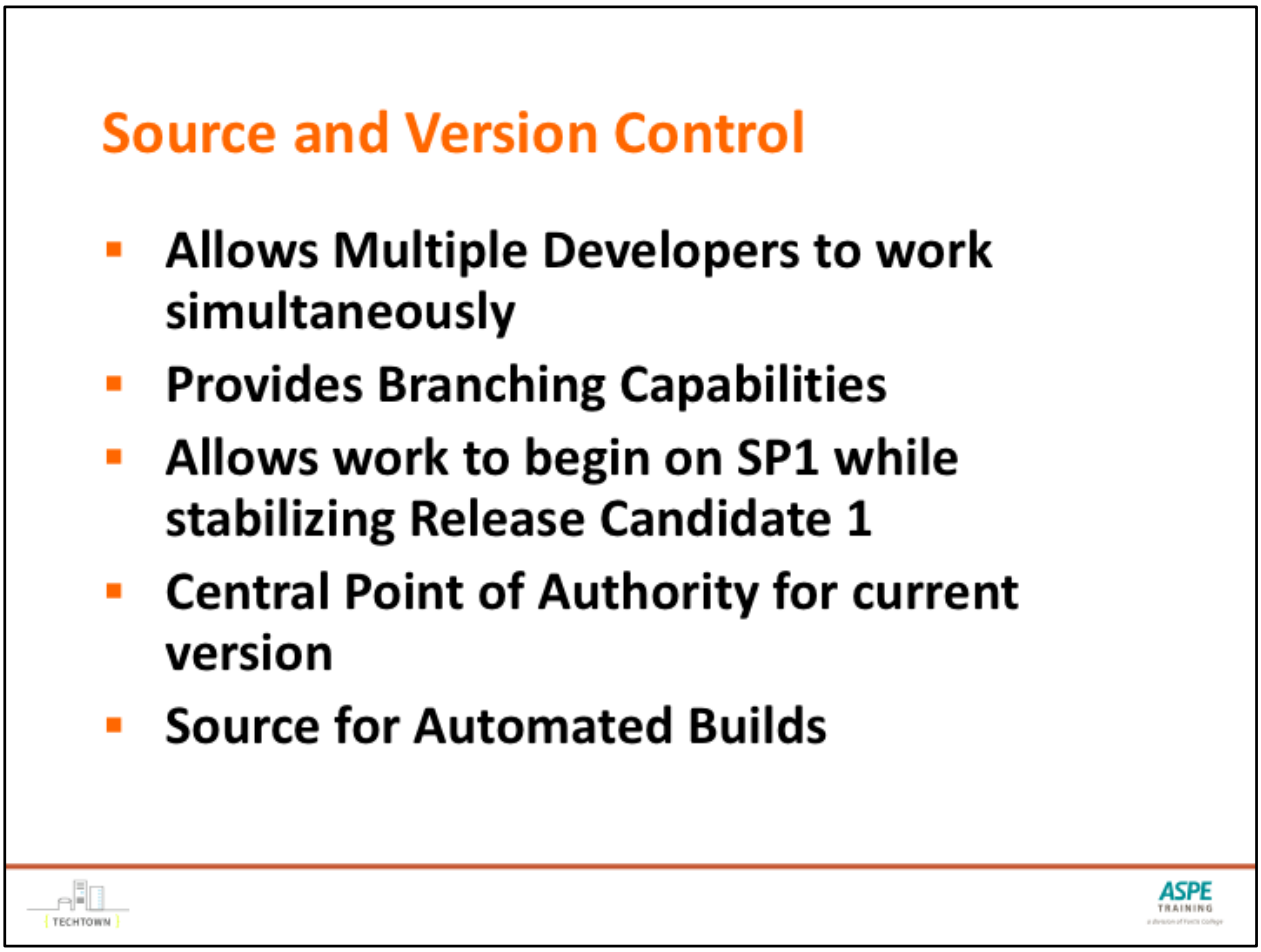
- Consistent Feedback for Developers
- Creating the Feedback loop
- Code Validation

Developers improve from consistent feedback
When developers get immediate feedback on code that they submitted recently, the code will be fresh on their minds, and they will be able to find the questionable code, identify the error, and learn from the mistake.

- Avoid the “What code, When?” scenario
- Automated Testing
- Feedback now, not later

Cultural Shift for Development & Operations
Aligning objectives between Development and Operations teams will result in a productive feedback loop, and a necessary cultural shift that saves time and effort during the testing phase of development.
Considerations that may require cultural change:

- Release Criteria
- Test environments
- Developers and Operations using the same tools



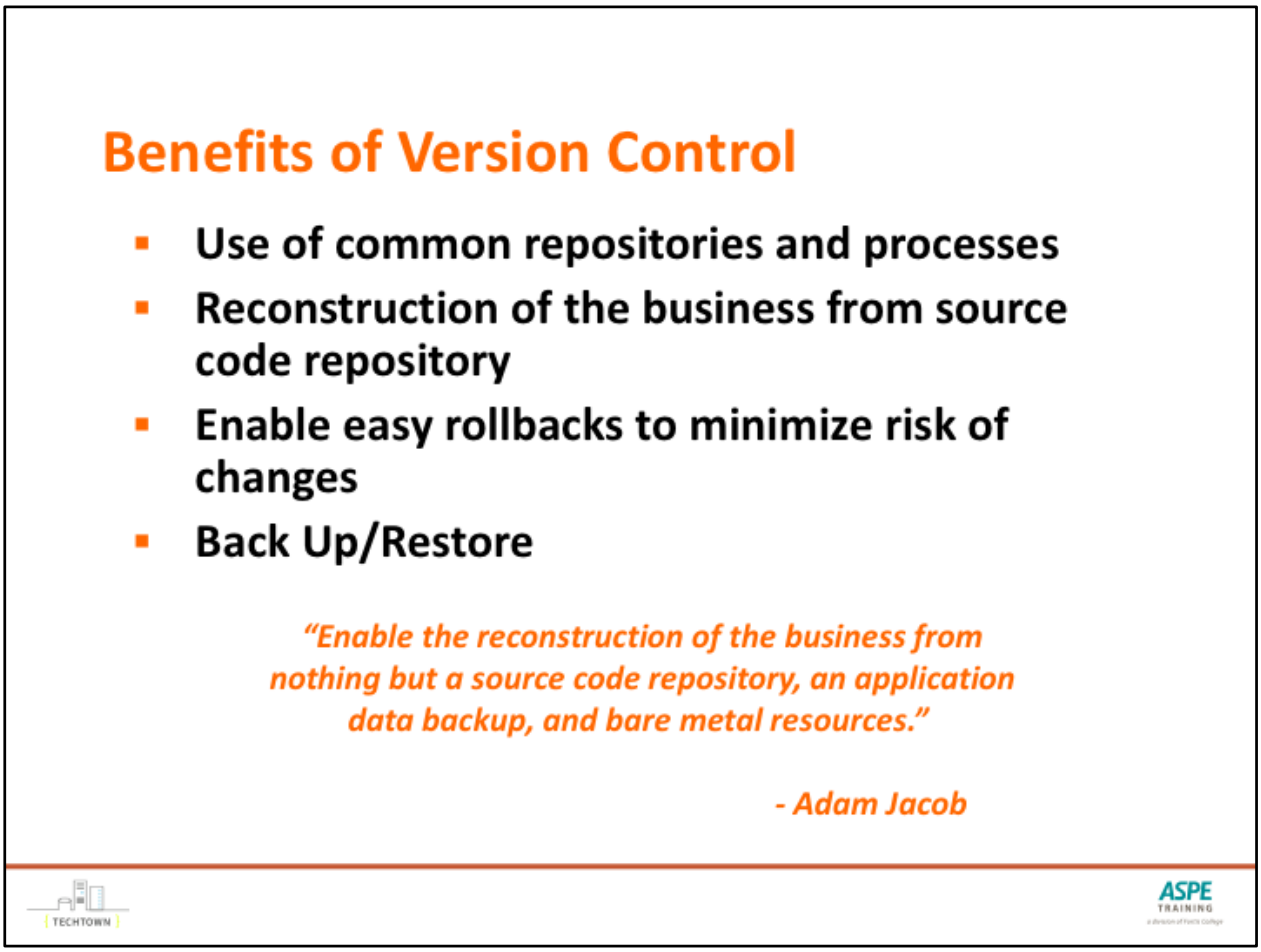
Version Control should achieve these things for you:

1. Record changes to files
2. Recall specific versions later
3. Enable Checked-in and reproducible
    - Application
    - Any deliverable artifacts
    - Configuration data
4. Enable the Team to:
    - setup environment from one source
    - duplicate the product from the repo
    - compile the product to demonstrate to a user
5. Enable Small, Frequent Commits
6. Dependency management
    - Build Binaries Only Once
    - Version control is not just one tool, it's a strategy.

- Anti-patterns

Common Source Control and Versioning Tools

- Team Foundation Server Version Control
- GIT/GITFLOW
- GITHUB
- SubVersion



Use common version control repositories and processes for both software and infrastructure.

- **Failure to place everything in version control can lead to reduced quality, agility and efficiency.**
    - New (and current) developers wasting time setting up environments
    - Reduced testing coverage due to challenges creating the artifacts
    - Loss of opportunities due to lack of easily demonstrable value
    - Version

- *Enable the reconstruction of the business from nothing but a source code repository, an application data backup, and bare metal resources."*

- **Track changes**
    - Track changes in version control
    - Enable easy rollbacks to minimize risk of changes

- **Back Up/Restore**
    - Frequent, automated, retrievable backups
    - Frequently restore and verify backups!

- **Disaster recovery benefits from Devops practices**
    - Reuse deployment automation to rebuild application in disaster scenario
    - Rebuilding the application from a single source in version control
    - Automating and limiting dependencies
    - Storing data and configuration in source control
    - Practicing and anticipating failure

Benefits of Version Control (continued)

Disaster recovery benefits from Devops practices

- Reuse deployment automation to rebuild application in disaster scenario
- Rebuilding the application from a single source in version control
- Automating and limiting dependencies
- Storing data and configuration in source control
- Practicing and anticipating failure

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Test Driven Development

- AKA Test “First” Development
- Create Unit Test before code is written
- Ensures all code has an associated test
- If there is a defect in the solution there is a missing or incomplete test
- Create a test for the defect before fixing the defect
- Build quality in early – don’t wait until release

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Finding & fixing Code Base Defects
Finding and fixing errors can help to bring code base up to release level standards during the testing phase of development. Once full regression testing is in place, time spent on fixing functionality should be reduced significantly.

Automate Daily Regression Testing
In large organizations, debugging deployment will be a difficult task. Teams will need automated processes that find & identify deployment issues, isolate them, and fix them before testing for code defects.

Add Code without breaking existing functionality

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Finding & Fixing Code Base Defects

- Bringing the Code Base to Release Quality
- Continuous Deployment Techniques
- Multiple daily Check-ins

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Automate Daily Regression Testing

- Automating daily testing
- Decreasing production time
- Working towards Frequent Releases

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

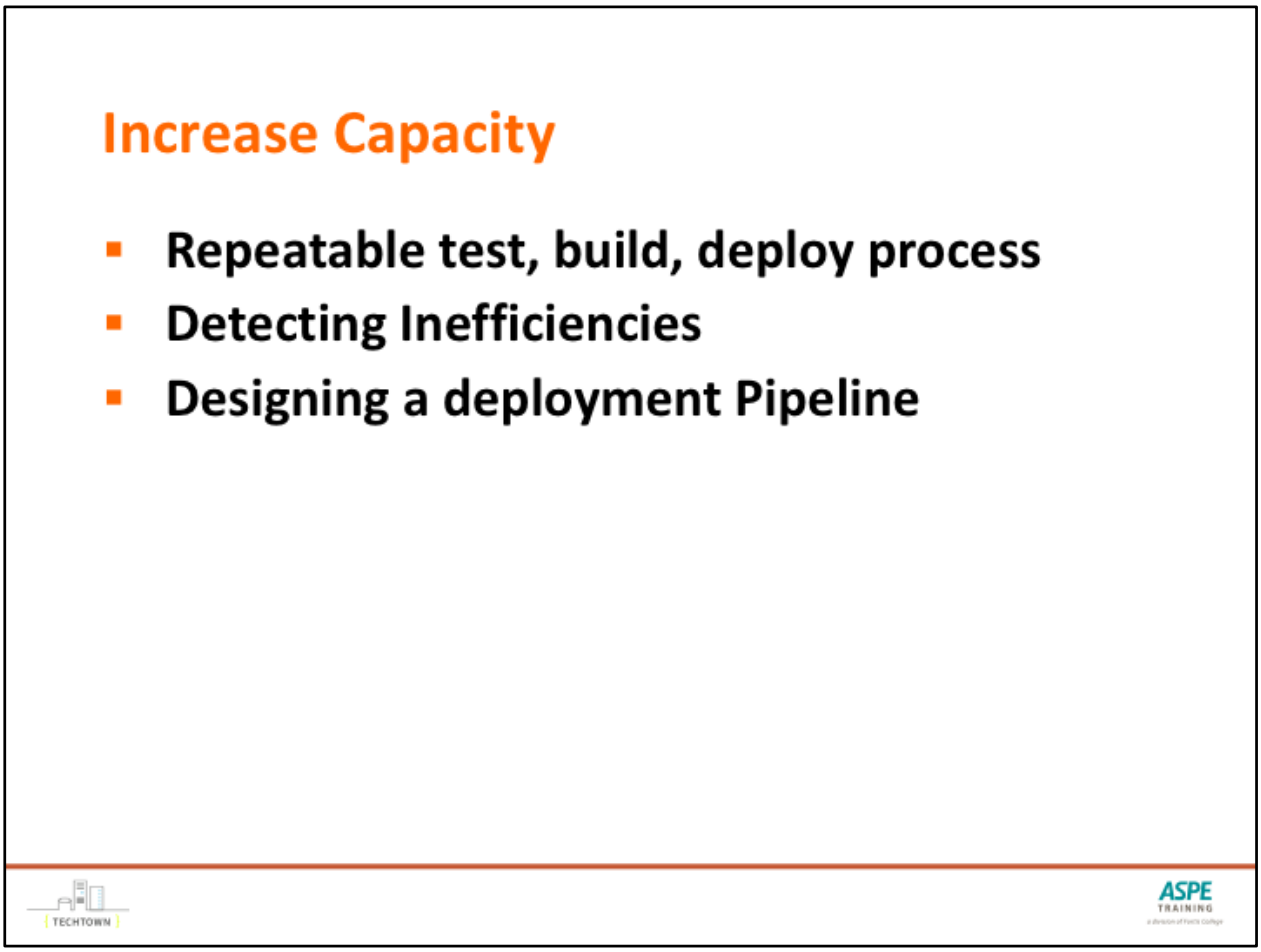
Add Code Without Breaking Existing Functionality

- Localize broken Code
- Isolate Deployment Issues
- Decrease Manual testing

ASPE TRAINING

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Repeatable test, build, deploy process
For organizations that have large groups of developers working together on a leveraged code base, or multiple different applications that need to work together

Detecting Inefficiencies
For traditional companies, de-bugging the deployment process can be complicated, especially if teams are deploying to multiple servers. Creating a deployment process that quickly isolates deployment issues is effective because it differentiates these concerns from bad code.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Repeatable Test, Build, Deploy Process

- Development and Operations working together
- Aligning common objectives
- Applying DevOps principles at Scale

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Detecting Inefficiencies

- Providing Immediate Feedback
- Feedback weeks later has limited value
- Delivering Value to the Customer

ASPE TRAINING

NOTES:

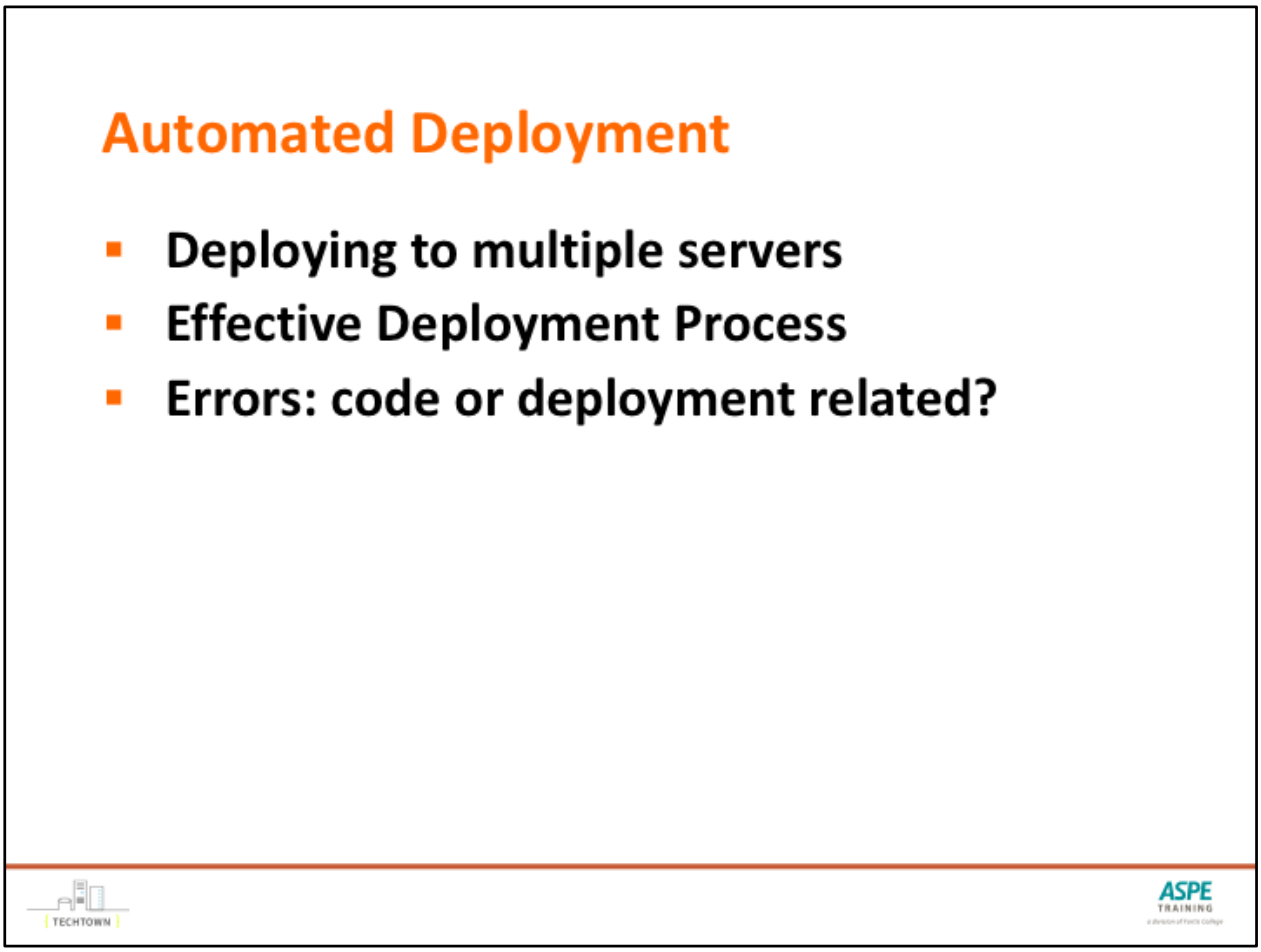
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Designing a Deployment Pipeline

- Designing a pipeline specified to your organization
- If it's labor intensive, it's probably wrong
- Structured, repeatable processes

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Deploying to multiple servers
Deploying to multiple servers is difficult, but once the process is automated it's less prone to mistakes & user error. Automated deployments capture and store the knowledge of how to release your software in the system, as opposed to in a developer's brain.

Debugging the Deployment Process
Once offending code errors have been separated from deployment issues, it makes the debugging that much easier and quicker.

Errors: code or deployment related?
We definitely want to know what is deployment related, and what is bad code that snuck through the gated check-in.

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Deploying to Multiple Servers

- Organization Size Matters
- Removing duplication of Work

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Debugging the Deployment Process

- Isolating deployment issues
- Finding code defects in a complex system

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Errors: Code or Deployment Related?

- System Test Failures
- Testing deployment before testing code

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

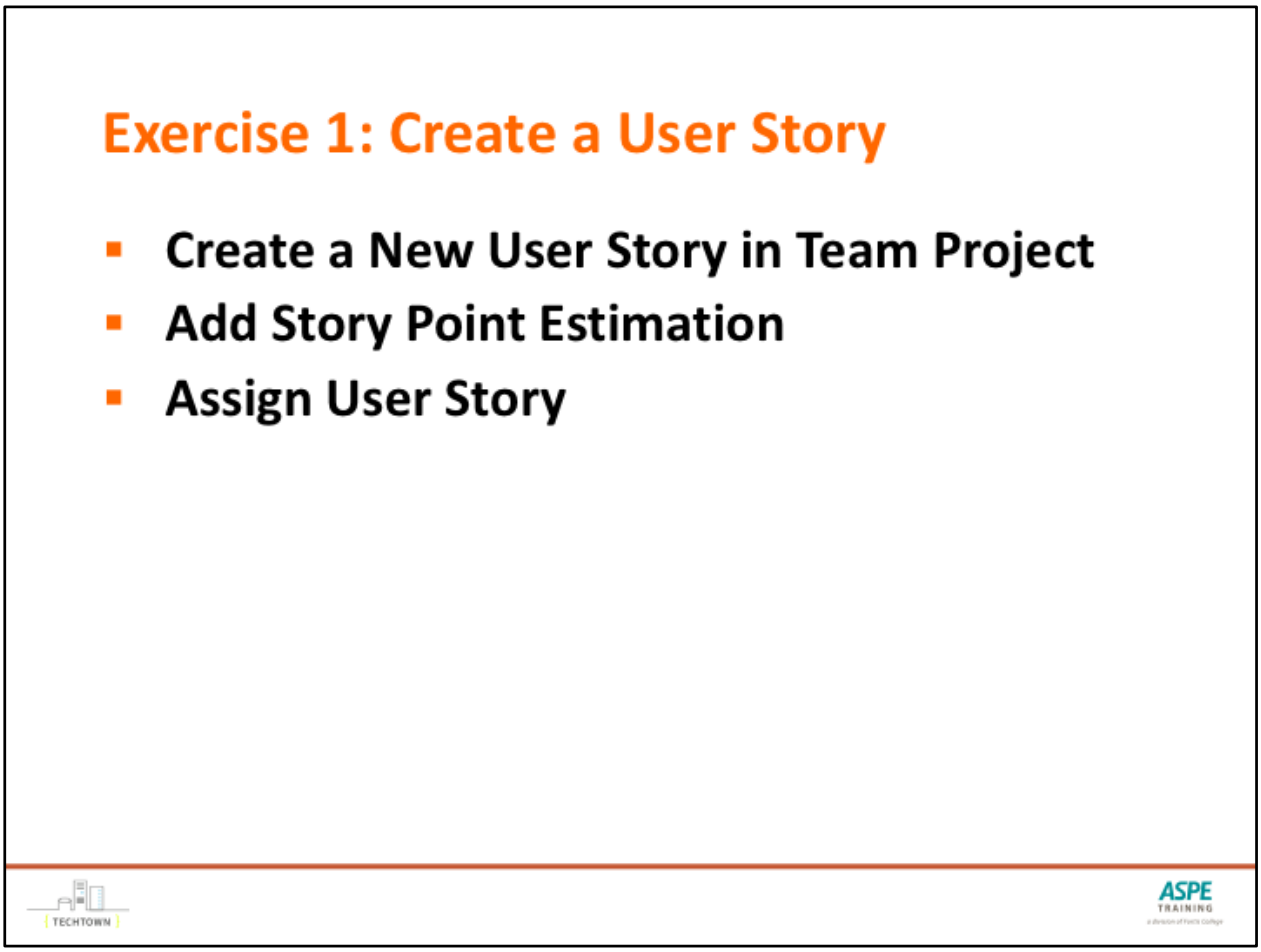
Lab: Be Agile With Team Services

- Exercise 1: Create a User Story
- Exercise 2: Define Acceptance Criteria
- Exercise 3: Define TDD Test Tasks
- Exercise 4: Define TDD Implementation Tasks

Lab guide: Page 40 to 51

NOTES:

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 1 Add User Stories to Product Backlog
In this exercise we will add 4 customer stories and 3 service provider stories with their associated acceptance criteria.

1. Select **Work** from the **VisualStudio.com** menu
2. On the board that appears click the **New** item button
3. Enter the following User Story in the Text box that appears and press enter:

Customer Story 1
As a customer, I need to register so that I can connect with service providers
Note: Can you think of other User Stories for this scenario?

4. Click on the link text of the user story
5. When the user story editor appears Assign the user story to yourself
6. Add a 2 to the story points field

Repeat steps to add the User Stories below

(*continued on next page)*

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Customer Story 2
As a customer, I need login so I can create reservations

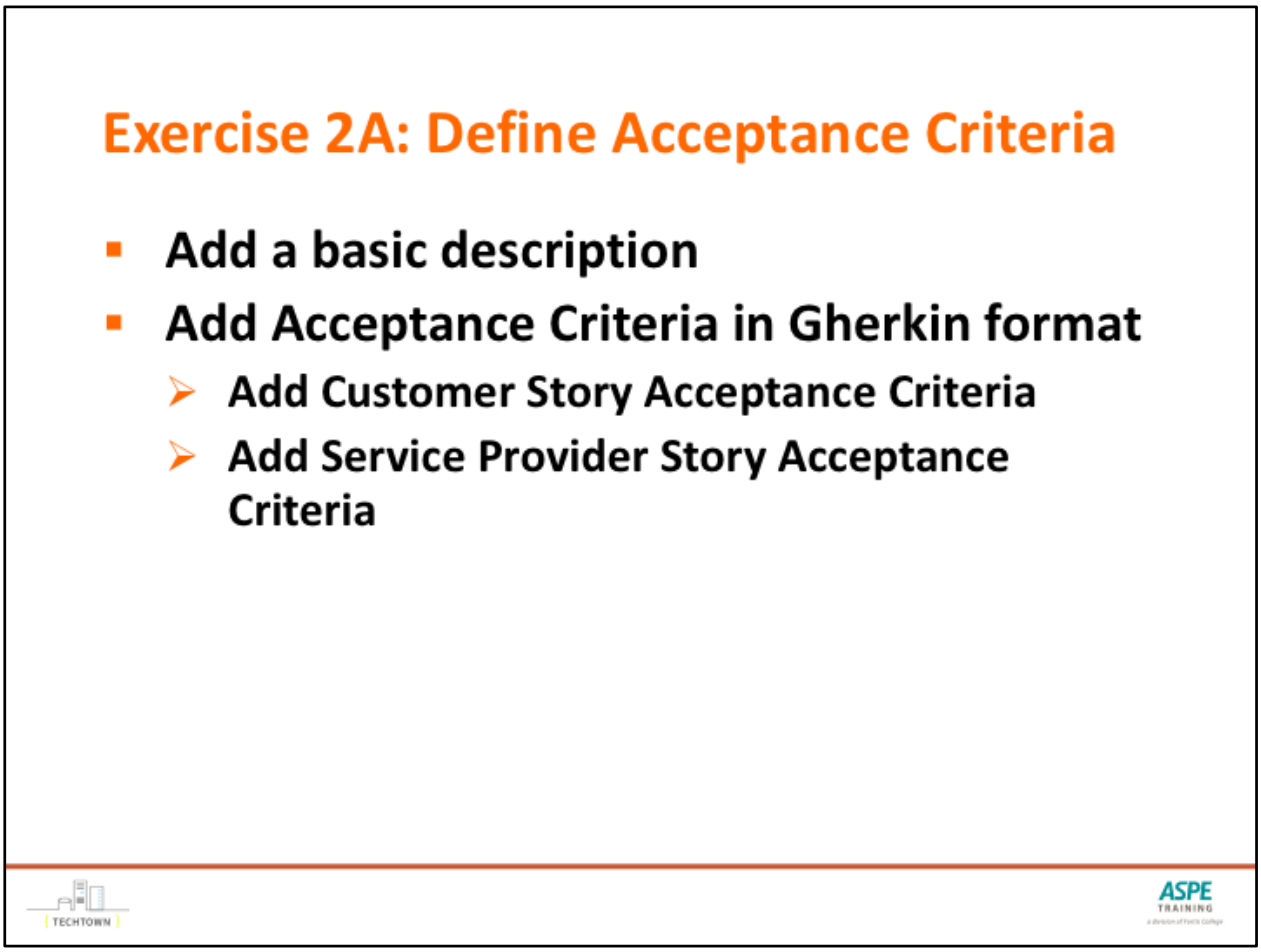
Customer Story 3
As a customer I need to enter the service I wish to purchase so that I can be connected with many service providers

Customer Story 4
As a customer, I need enter the price I wish to pay so that I can be connected with many service providers in that price range

Service Provider Story 1
As a Service Provider, I need to register my company so customers can connect with me

Service Provider Story 2
As a service provider, I need to specify each service I provide with range of prices so only customers in my price range are connected with me

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 2 Add Acceptance Criteria to User Stories

In this exercise we will add Acceptance Criteria to the 4 customer stories and 2 service provider stories added in the previous exercise.

1. Select Work from the VisualStudio.com menu
2. Once the Board appears click the New Item button
3. Enter the following User Story in the Text box that appears and press enter:

Customer Story 1

As a customer
I need to register
So that I can connect with service providers

4. Click on the link text of the user story
5. When the user story editor appears enter the following **Acceptance Criteria:**

    Given a valid registration information (First Name, Last Name, Phone Number, Zip Code)
    When I attempt to Register
    Then I am taken to the Registration Confirmation page
    Given invalid registration information (First Name, Last Name, Phone Number, Zip Code)
    When I attempt to Register : Then an error is displayed indicating "Invalid Registration Details"
    (Specific Details noted with *)

Note: Can you think of other Acceptance Criteria for this story?

Repeat steps to add the Acceptance Criteria below

(*continued on next page*)

Customer Story 2
As a Customer
I need login
So that I create reservations

Customer Story 2 Acceptance Criteria
Given a valid username and password
When I attempt to log in
Then I am taken to the profile page

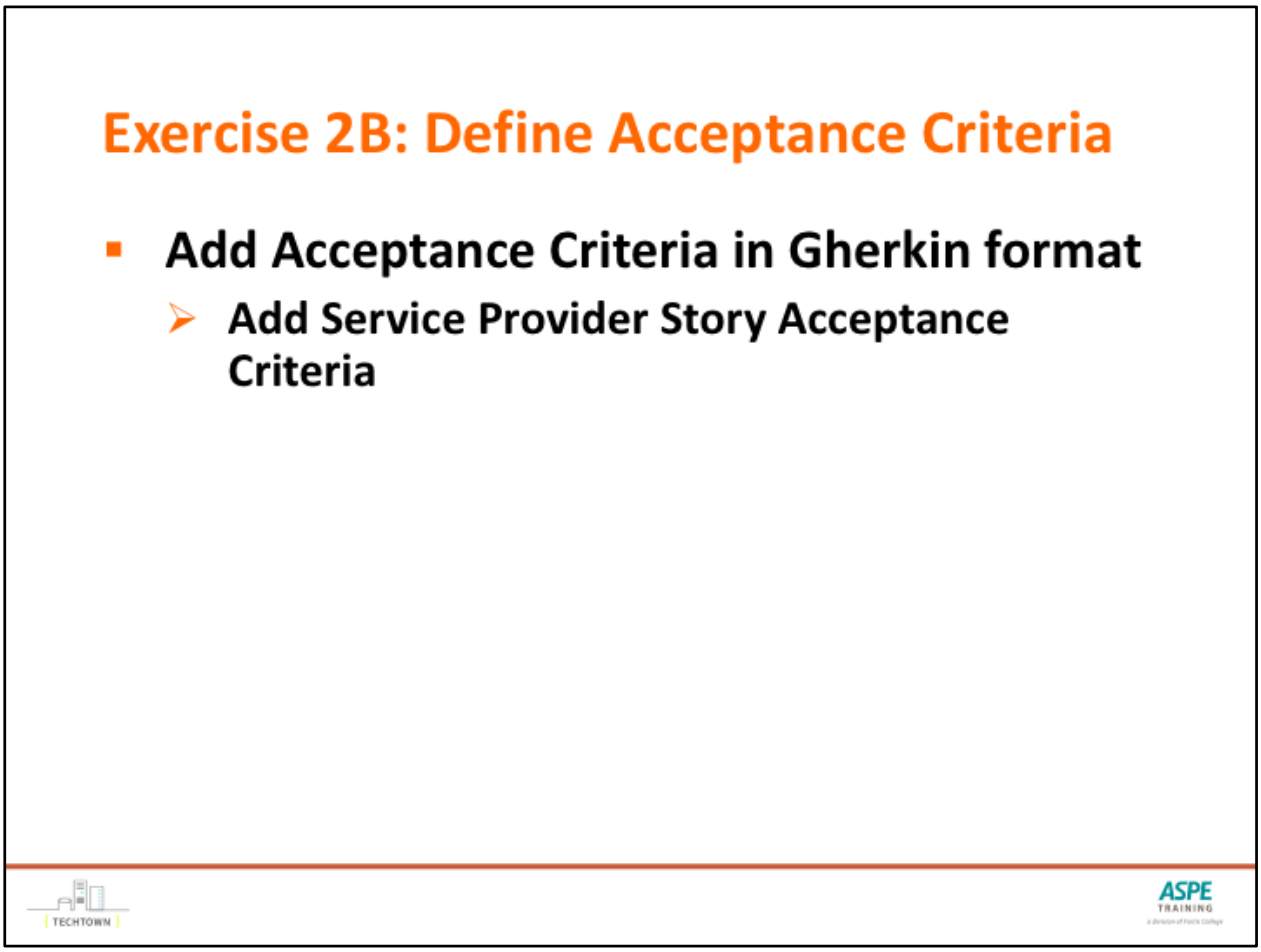
Given an invalid username or password
When I attempt to log in
Then an error is displayed indicating "invalid username or password"

Customer Story 3
As a Customer
I need enter the service I wish to purchase
So I can be connected with many service providers

Customer Story 3 Acceptance Criteria
Given a valid Service Name and details
When I submit my service request
Then I am taken to the price and confirmation page

Customer Story 4
As a Customer
I need enter the price I wish to pay
So that I can be connected with many service providers in that price range

Customer Story 4 Acceptance Criteria
Given a Price within service providers defined range of prices this service
When I submit my request
Then I am taken to the confirmation page



Service Provider Story 1
As a Service Provider
I need to register my company
so that customers can connect with me

Service Provider Acceptance Criteria
Given a valid username and password
When I attempt to log in
Then I am taken to the profile page
Given an invalid username or password
When I attempt to log in
Then I error is displayed indicating "invalid username or password“

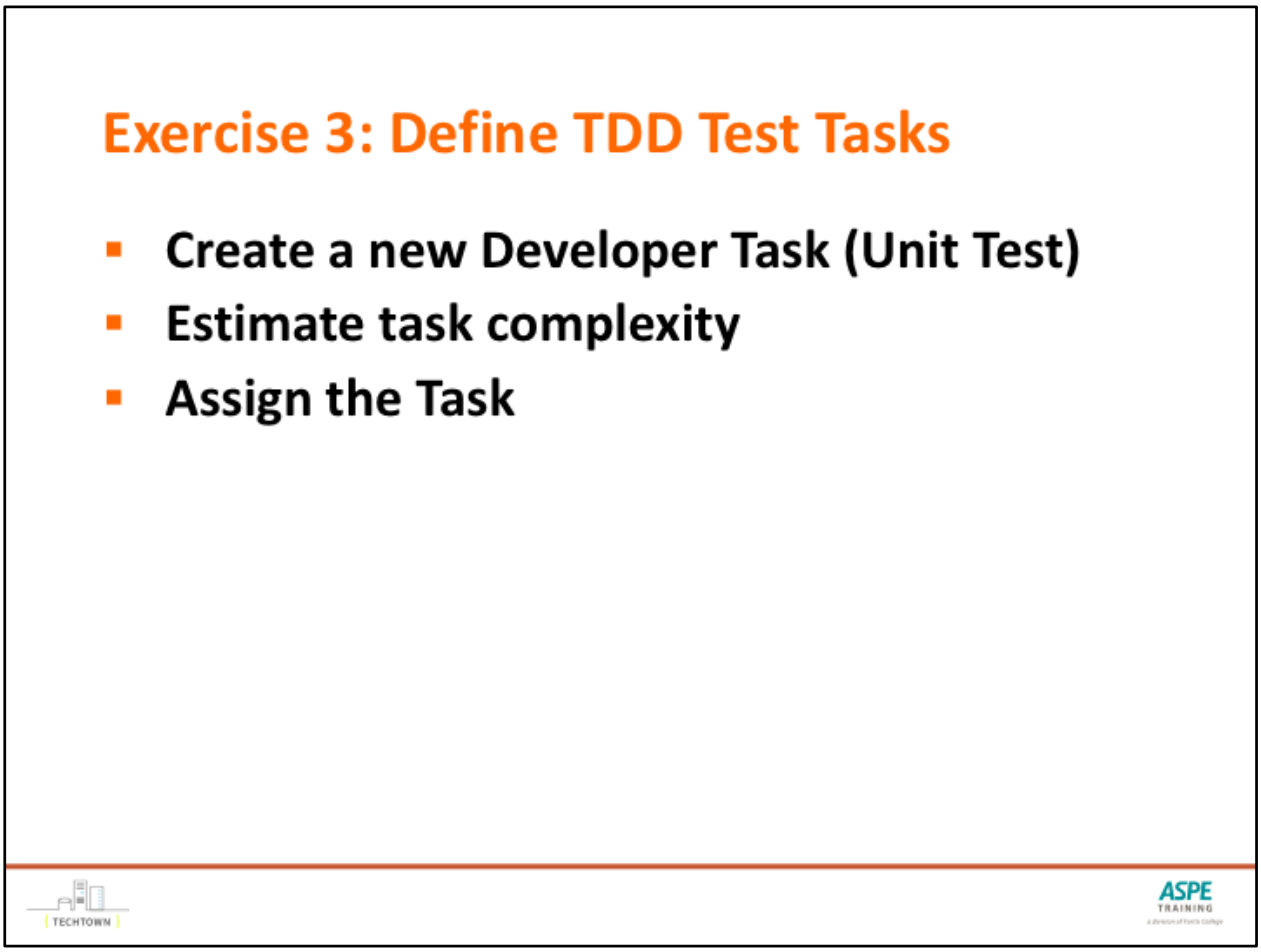
(continued on next page)

Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved

Service Provider Story 2
As a service provider
I need to specify each service I provide with range of prices
So that only customers in my price range are connected with me

Service Provider Story 2 Acceptance Criteria
Given a valid service name and price range
When I submit my service offering
Then I am taken to the confirmation page

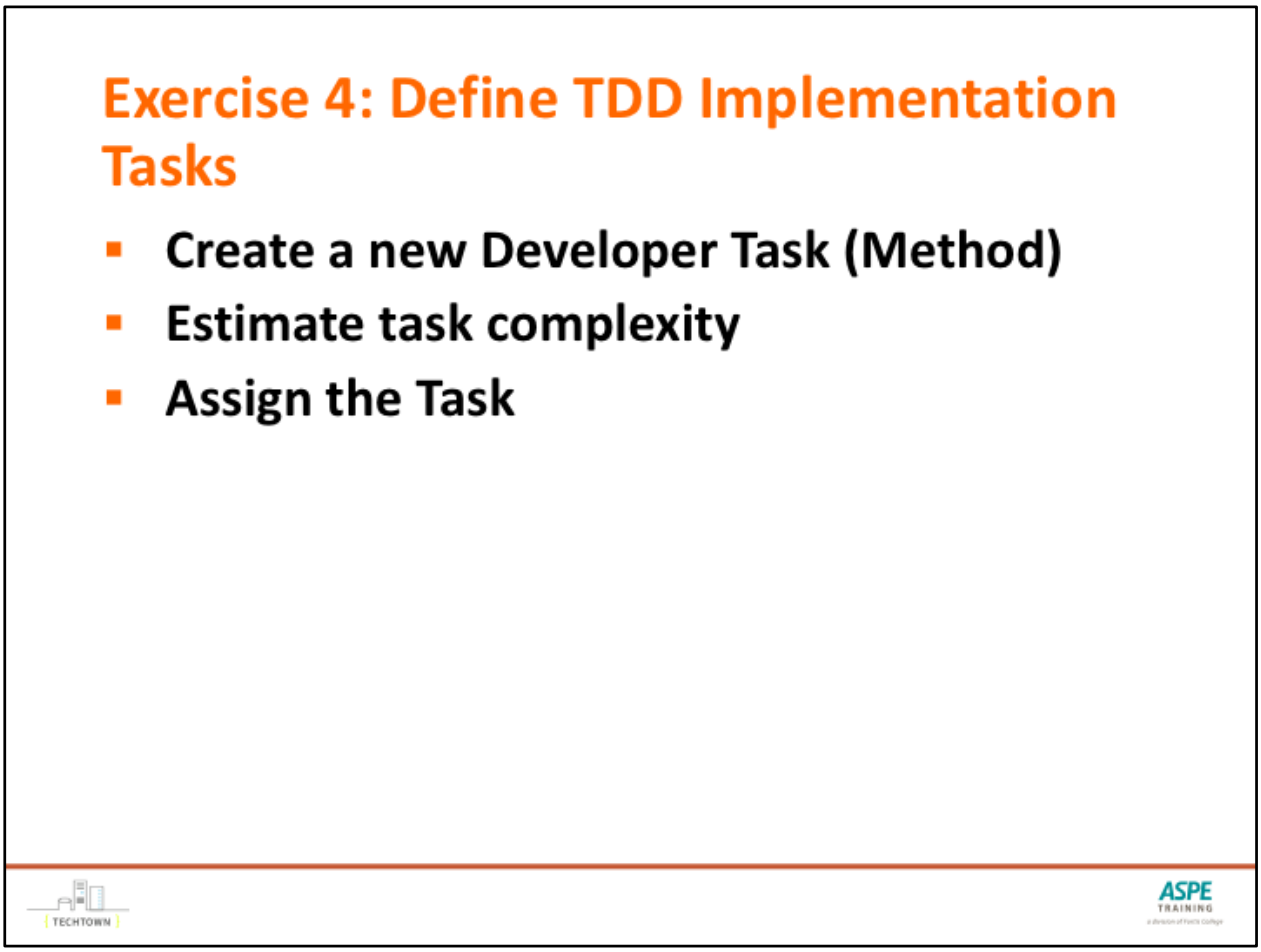
Continuous Delivery © 2019 ASPE | Licensed for Classroom Distribution Only | All Other Rights Reserved



Exercise 3 Add Delivery Team Tasks to User Stories
In this exercise we will add Delivery Team Tasks (Test details) to our user stories and surface any currently undocumented dependencies and risks that may impede the team's progress during the sprint.

1. Select Work from the VisualStudio.com menu
2. Once the Board appears click on the link text of **Customer Story 1**
3. When the user story editor appears click Add Link in the Related Work section to the right.
4. In the Add Link menu choose New Item
5. In the Add Link dialog enter "Implement Unit Test of ASP.Net MVC Registration" as the Title for the task and click OK.
6. Enter 1 in the Original Estimate field under **Effort (Hours)**
7. Ensure that the task is assigned to you then click **Save & Close**

8. Click on the link text of **Customer Story 2**
9. When the user story editor appears click Add Link in the Related Work section to the right.
10. In the Add Link menu choose New Item
11. In the Add Link dialog enter "Implement Unit Test for ASP.Net MVC Login" as the Title for the task and click OK.
12. Enter 1 in the Original Estimate field under **Effort (Hours)**
13. Ensure that the task is assigned to you then click **Save & Close**



Exercise 4 Add Delivery Team Implementation Tasks to User Stories

In this exercise we will add Delivery Team Tasks (Implementation details) to our user stories and surface any currently undocumented dependencies and risks that may impede the team's progress during the sprint.

1. Select Work from the VisualStudio.com menu
2. Once the Board appears click on the link text of **Customer Story 1**
3. When the user story editor appears click Add Link in the Related Work section to the right.
4. In the Add Link menu choose New Item
5. In the Add Link dialog enter "Implement ASP.Net Membership" as the Title for the task and click OK.
6. Enter 1 in the Original Estimate field under **Effort (Hours)**
7. Ensure that the task is assigned to you then click **Save & Close**

8. Click on the link text of **Customer Story 2**
9. When the user story editor appears click Add Link in the Related Work section to the right.
10. In the Add Link menu choose New Item
11. In the Add Link dialog enter "Create a new ASP.Net MVC Project with Individual User Accounts template" as the Title for the task and click OK.
12. Enter 1 in the Original Estimate field under **Effort (Hours)**
13. Ensure that the task is assigned to you then click **Save & Close**