Automating Builds & Deployments

Benjamin Day
TRAINER | COACH | DEVELOPER

@benday www.benday.com

Overview

- Why automated builds + DevOps?
- Builds vs. release pipelines
- TFS2017 build system
- Different types of build triggers
- Customizing the build process
- TFS extensions / build activities
- Installing the build agent
  - Windows
  - Linux
- Agent Capabilities

Automated Builds: “Why do I care?”

The
“works on my box”
problem.

"Well, it works on my box."

Works on a developer's machine

Doesn't work somewhere else

- Configuration differences
- Subtle code differences
- Version control issues

Integration issues

Annoying during development

Catastrophic when going to production

Automated builds & automated deployments

An automated build is the start of your DevOps awesomeness.

Automated Builds

Checks & balances

Build server keeps you honest

- Unbiased judge
- Not the developer's machine

Is the build broken?

Compiles on the build server?

- No → Build is broken

Unit tests pass on build server?

- No → Build is broken

DevOps & Automation

Automated builds run without human involvement

Humans are slow

Humans make mistakes

Humans are inconsistent

Automation = consistency, repeatability

TFS Builds vs. TFS Releases

Two parts that work together

TFS Builds compile the code

- Integration
- Turn code into an application
- Get *ready* to deploy

TFS Releases deploy the application

- Understand deployment environments
- Build, Test, Stage, Production, etc.

Much more on TFS Releases later

Separating “build” from “release” changes your testing process.

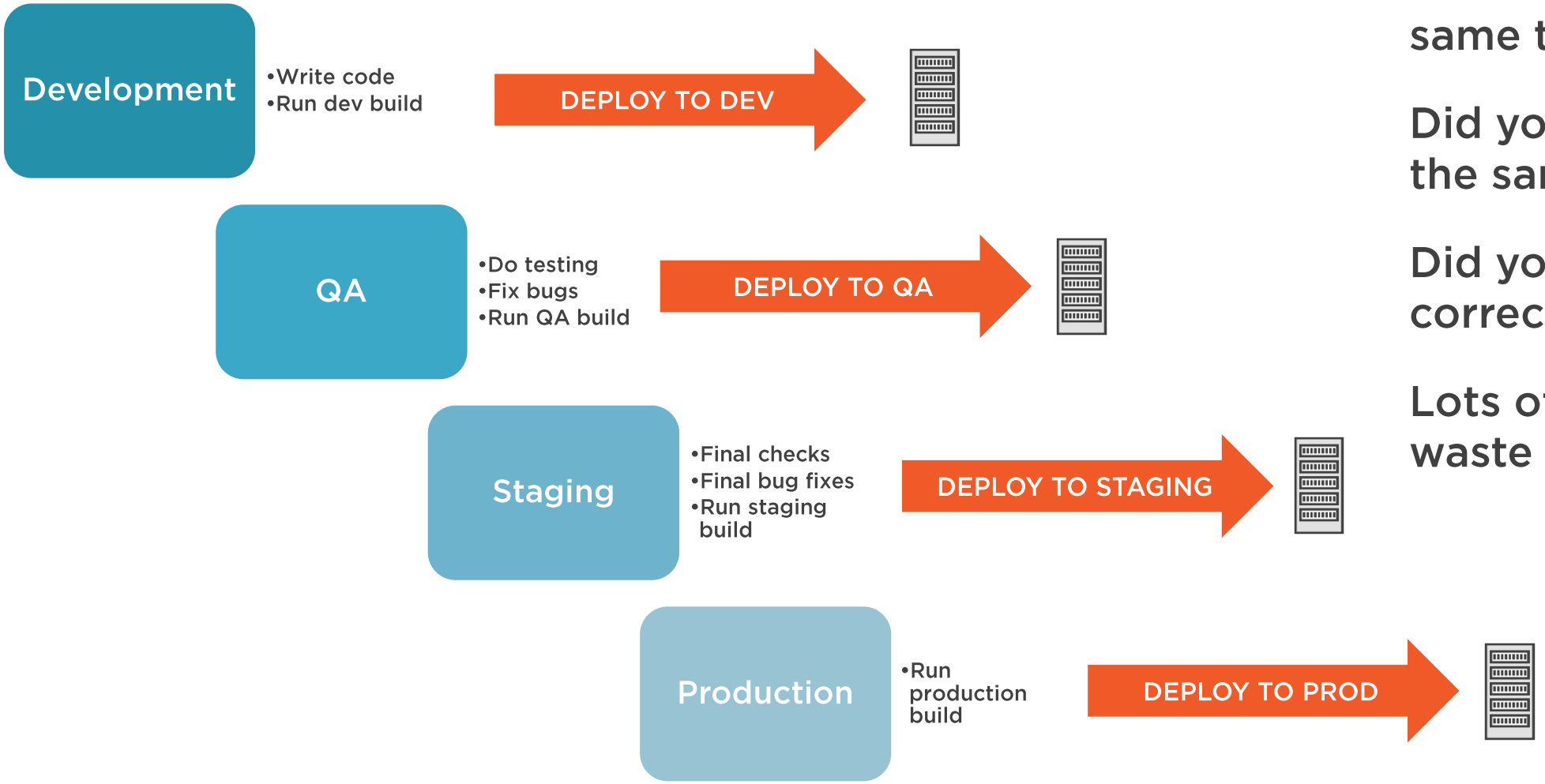
Separate Build from Release / Deploy

Compile once and *only* once

Deploy to many environments

Always test the same compiled application

Phases / Environments != Branches



Never deployed same thing twice

Did you really test the same thing?

Did you merge correctly?

Lots of merging → waste

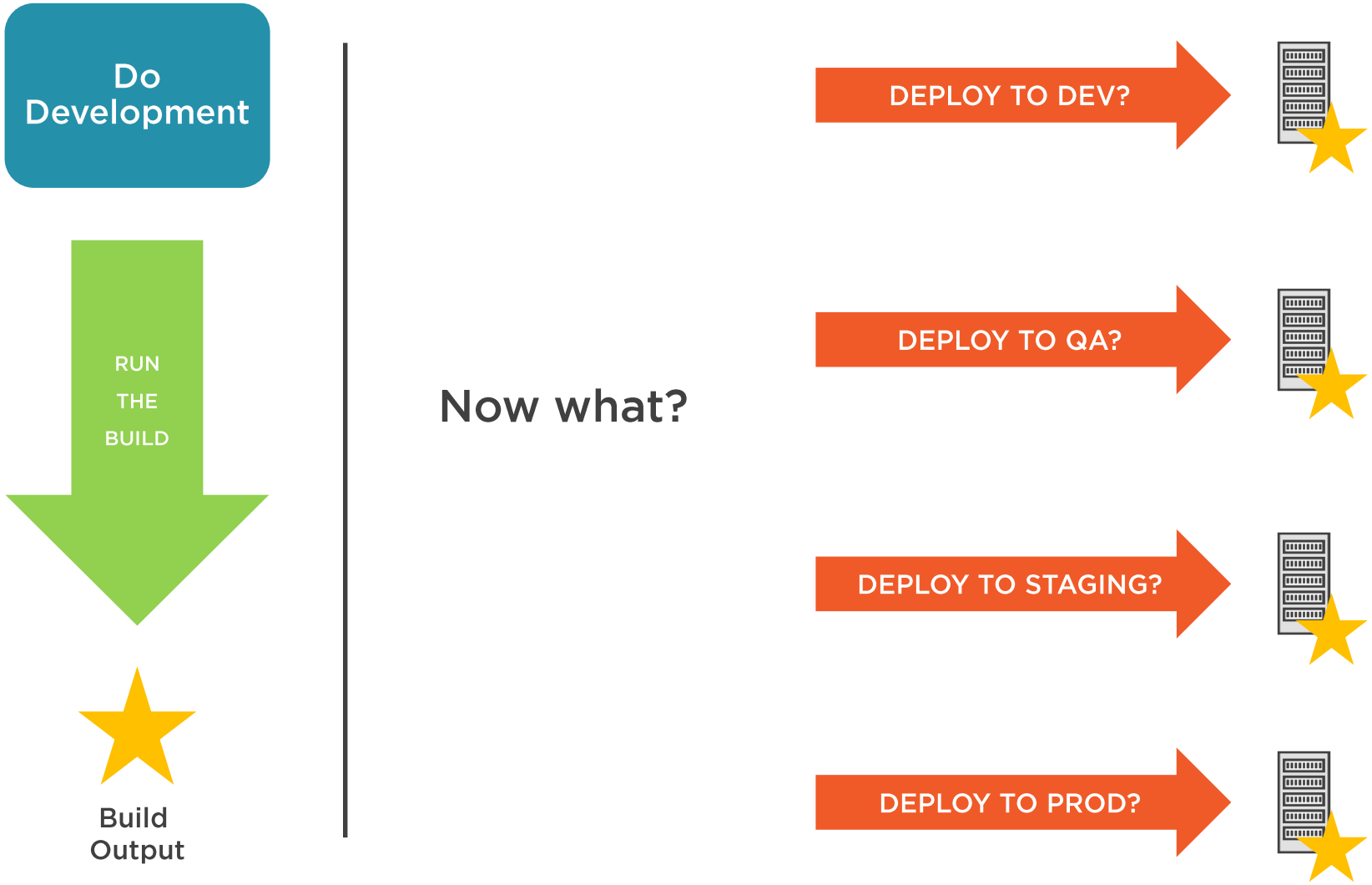
Separate Build from Release / Deploy

Compile once and *only* once

Deploy to many environments

Always test the same compiled application

One Build, Multiple Environments



Always deployed same build output

Always tested the same thing

No merges required

No need to go through all environments

The “works on my box” problem can also be caused by deployment issues.

Deployment Annoyances

"Works on my box"

Deployment details

- IIS configuration
- File & folder permissions
- Database connection strings
- Service endpoint URLs
- Database deployments
- Database lookup data

Deployments are traditionally high-risk

Infrequent Deployments

Why?

- Painful
- Manual effort
- Takes a lot of time
- Stuff breaks unpredictably → fire drills

Do painful stuff as infrequently as possible

Infrequently means you forget...

- ...how to deploy
- ...how to configure

Customers don't get new stuff that often

- Slow feedback
- Slow fixes
- Defensive posture

Configuration Steps: Manual or Automated?

Manual	Automated / Scripted
Forgotten steps	It's the same every time
Troubleshooting problems	It works or it doesn't work
Similar but not identical	Didn't work? Fix the script.
Deployment is a Constraint	Script is version controlled
	Can be called from an automated build

What happens if your build is automated and your deploy is automated?

Builds and Deploys Are Automated

Not painful to integrate your team's code

Not a big deal to do a deploy

Little or no human involvement

If a deploy isn’t a big deal, then what’s stopping you from doing them more often?

At its core, DevOps is about shortening feedback cycles.

Shorter Feedback Cycles in DevOps

Show a feature, get feedback

Deploy a feature, see if it's popular

Take the feedback, improve the product

Make the product less awful as soon as possible

Repeat the process

If a deploy isn’t a big deal,
then what’s stopping you
from doing them
more often?

What’s stopping you from doing them *A LOT* more often?

Automated builds are the start of your DevOps awesomeness.

Next up:
Create an automated build
in Team Foundation Server

Demo

Create a TFS build

.NET Standard

Run unit tests

Build trigger types

Next up:
Create a build for .NET Core

Demo

Create a TFS Build

.NET Core

Next up:
Builds on the dashboard

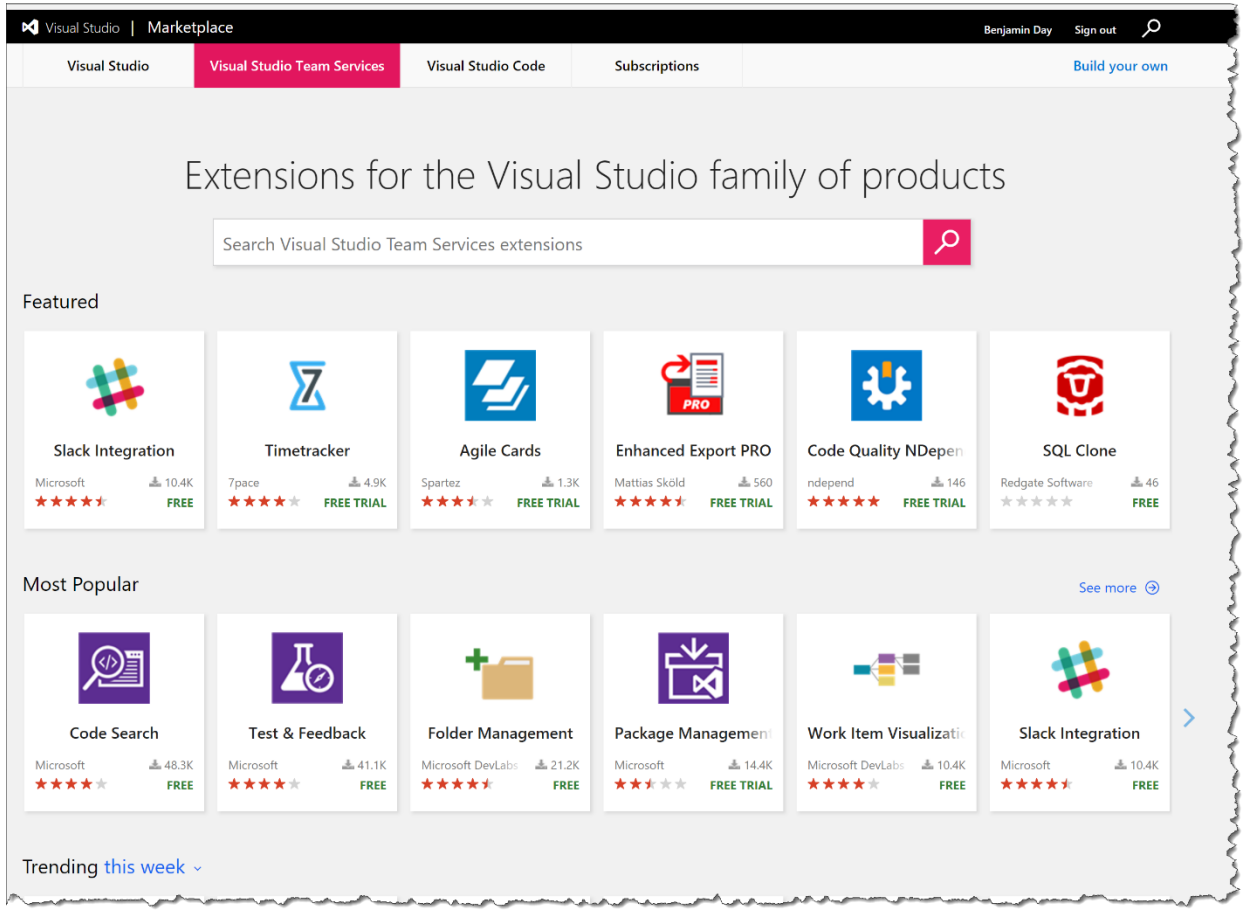
Demo

Builds and project dashboards

Next up:
TFS Extensions & Build Activities

TFS is extensible.

Visual Studio Marketplace



https://marketplace.visualstudio.com/

TFS Extensions

Download from the Marketplace

Write your own

- Publish to the world
- Private

http://*{server-name}*:8080/tfs/_gallery/manage

Demo

Find an extension in the marketplace

Install the extension in TFS

Use the extension in a build

Next up:
Create & deploy a
build agent

Team Foundation Server Agent

Does the work for TFS Build & Release

Available on Windows, Linux, & MacOS

Installing the
TFS Agent is easy.

TFS 2017 Install Guide

Team Foundation Server 2017 Installation Guide Page 1 of 118

Team Foundation Server 2017
Installation Guide

BDC Benjamin Day Consulting

Benjamin Day
benday@benday.com

v1.1.0
February 27, 2017

Copyright © 2017 – Benjamin Day Consulting, Inc. – www.benday.com
Printing or duplication is prohibited without author's expressed written permission.

https://www.benday.com/tfs2017installguide

Before You Start...

Authentication for the Agent to TFS

- TFS with Active Directory → {domain}\TFSBUILD
- Visual Studio Team Services (VSTS) → Personal Access Token (PAT)

TFS URL

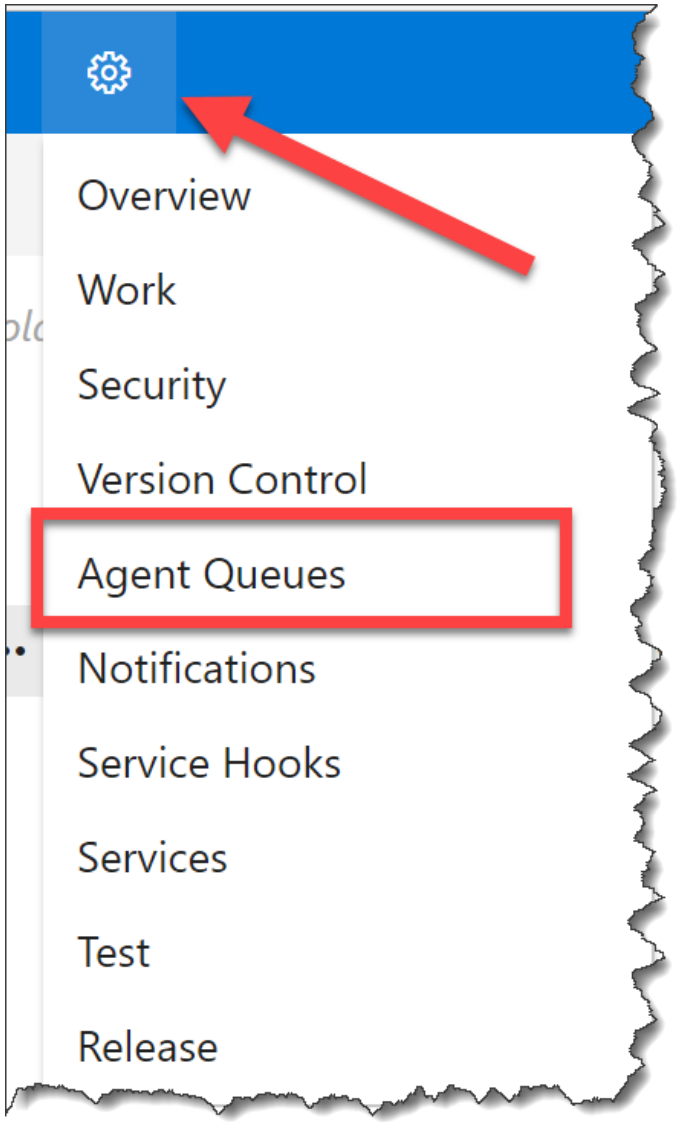
- http://{server-name}:8080/tfs

VSTS URL

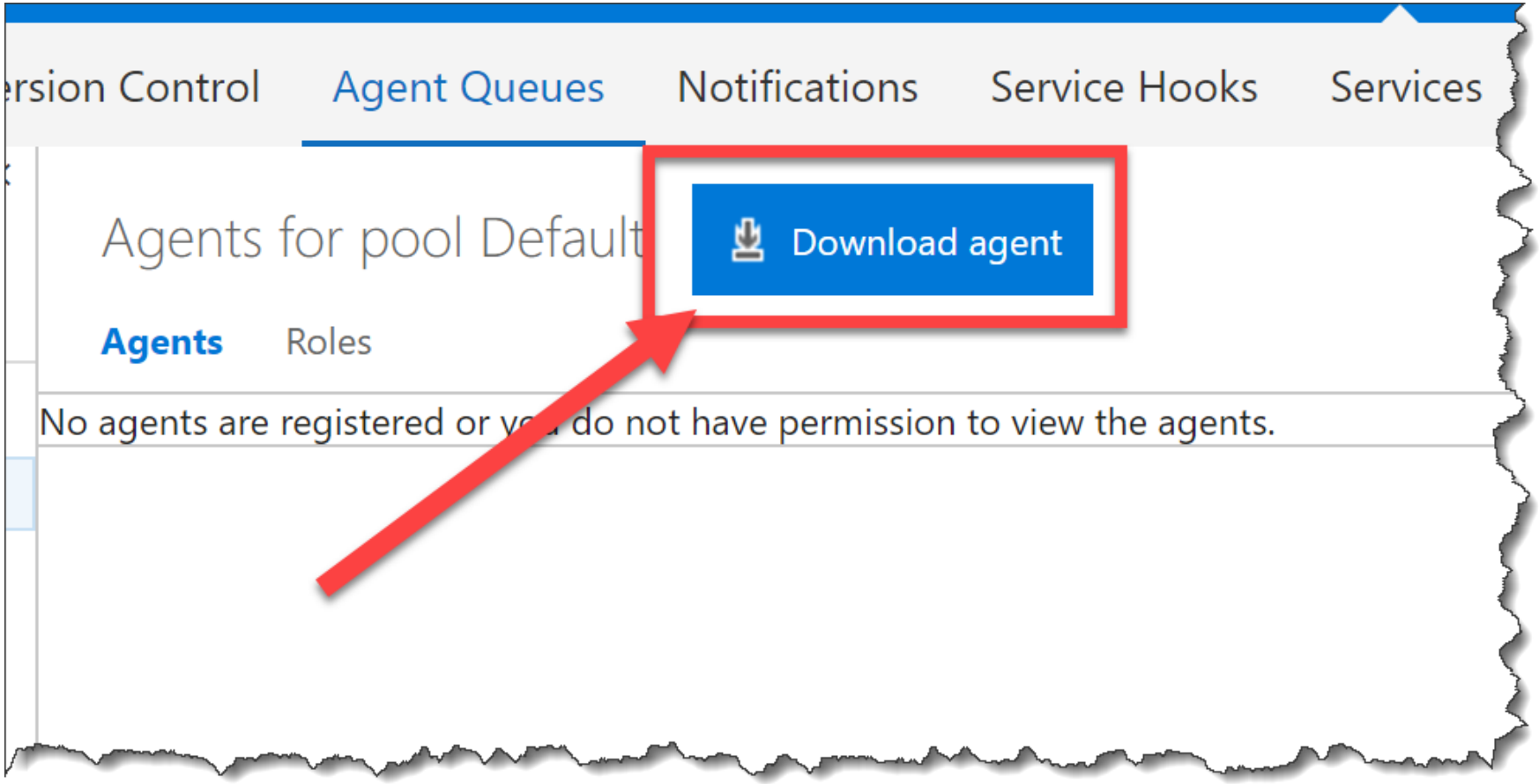
- https://{account-name}.visualstudio.com

Step 1:
Log into the server

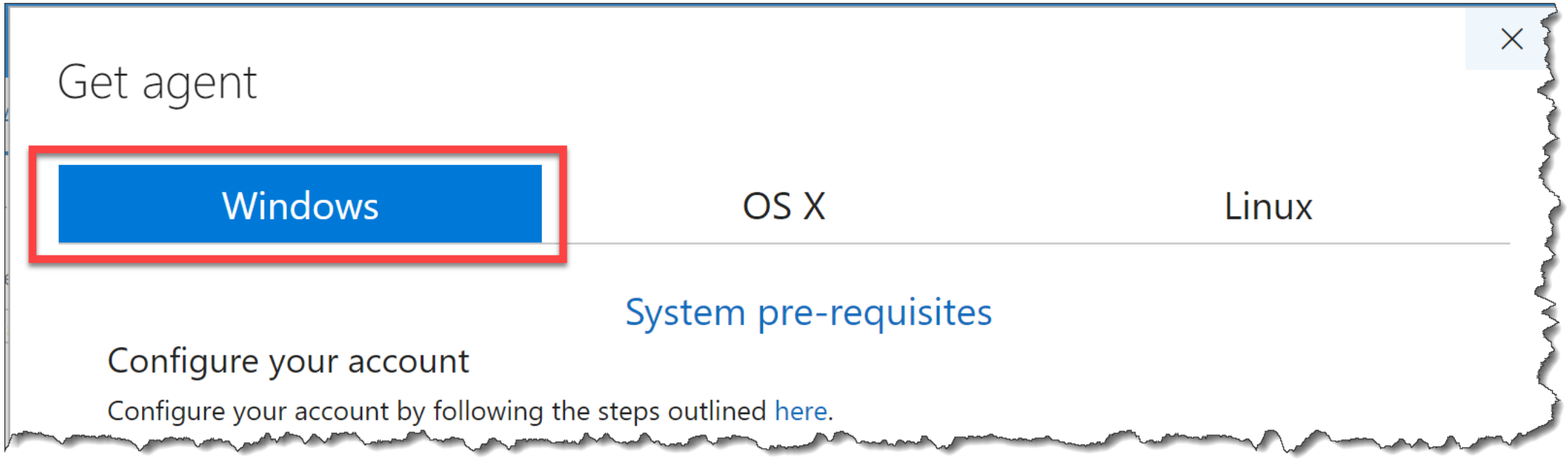
Step 2: Navigate to Agent Queues



Step 3: Click Download Agent



Step 4: Choose Operating System



Step 5: Download the Agent

Get agent

Windows | OS X | Linux

System pre-requisites

Configure your account

Configure your account by following the steps outlined here.

Download the agent

Download

Create the agent

```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-agent-win7-x64-
2.112.0.zip", "$PWD")
```

Configure the agent detailed instructions

```
PS C:\agent> .\config.cmd
```

Optionally run the agent interactively

If you didn't run as a service above:

```
PS C:\agent> .\run.cmd
```

Step 6: Follow the Instructions

Create the agent

```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-agent-win7-x64-
2.112.0.zip", "$PWD")
```

Configure the agent detailed instructions

```
PS C:\agent> .\config.cmd
```

Optionally run the agent interactively

If you didn't run as a service above:

```
PS C:\agent> .\run.cmd
```

That’s it.

Next up:
Agent Capabilities

TFS Agent Capabilities

Agent Capabilities

Define what an agent can do

Build definition demands

Match builds to agents

Demo

Configure build agent capabilities

Summary

- Why automated builds + DevOps?
- Builds vs. release pipelines
- TFS2017 build system
- Different types of build triggers
- Customizing the build process
- TFS extensions / build activities
- Installing the build agent
- Agent Capabilities

Next up:
Deploy database updates
from a build