Puppet Bolt Workshop Preparation Guide

Written by:
Paul Reed, Solutions Engineer
Toronto, Ontario, Canada
paul.reed@puppet.com

Revision Date: 2020-11-12

Table of Contents

1. What is Bolt?

Bolt is an open source, agentless, multi-platform automation tool that makes it easy to get started with or scale up infrastructure automation. Automate simple one-off tasks or create shareable, reusable workflows in any language, without agent software or prior Puppet knowledge.

2. What is a Bolt Workshop?

In the Bolt Workshop you'll learn how to execute simple commands across a fleet of Windows and/or Linux hosts, run existing scripts, turn them into Bolt tasks, and tie it all together into a Bolt plan to start automating your day-to-day tasks right away.

3. What is the purpose of this guide?

This guide will provide you with the necessary instructions to complete the required prerequisites for the Bolt Workshop. It is expected that all workshop participants will have completed this work prior to the start of the workshop. You'll need to use the same workstation to the workshop you used to complete the prerequisites!

4. Workshop Requirements

In order for the workshop to run smoothly, we're asking for a little bit of preparation work.

Before the workshop, you will need:

- **A Windows, Mac or Linux workstation** to use (choose any you're comfortable with). We will provide a Windows target host for the exercises during the actual workshop.
- **Microsoft VS Code** or another capable editor of choice for writing Scripts, Bolt Tasks and Bolt Plans. We will be using this tool specifically in the workshop, but if you are familiar with other tools and the use of integrated source control (git commands) you can use another preference. If you run into trouble using your own tools during the workshop, we won't have time to help you troubleshoot any related issues.
- **Internet access for your workstation.** If you are behind a firewall (either software, on your workstation, or a corporate hardware device), you will need outbound access enabled. We will be using remote targets that are "in the cloud" and you'll need to be able to access them!
- **Puppet Bolt** installed on your workstation. We'll show you how to download and install it in this guide, but make sure to have it up and running prior to the workshop!
- A burning desire to automate away soul crushing things!

5. Prerequisite Labs

In order to complete the preparation for the Bolt Workshop, the following labs are designed to be run locally from your workstation. You will need to use the same workstation for the actual workshop.

Please complete all labs in this guide prior to the start of your scheduled Bolt Workshop.

Lab 1: Installing Bolt

It is required to have Bolt installed and functional prior to starting the workshop. This will ensure that we get the maximum use of our time during the workshop to explore all the possibilities Bolt has to offer. You will only need to use Bolt from a single workstation during the workshop.

Bolt supports running from the following operating systems:

- Windows (MSI Based Installation or Chocolatey)
- MacOS (Homebrew Based Installation or MacOS Installer)
- RPM Enterprise Linux Installation (RedHat, CentOS, etc.)
- dpkg/apt-get Based Linux Distributions (Ununtu, Debian, etc.)

Note: Although running bolt from the list above is limited, the targets can be almost any OS or device.

We will cover the installation for Windows and MacOS in this guide. Further instructions for Linux or other installations can be found in the online documentation at:
https://puppet.com/docs/bolt/latest/bolt_installing.html

Windows MSI Based Installation

Use the MSI installer package to install Bolt on Windows.

1. Download the Bolt installer package from: https://downloads.puppet.com/windows/puppet6/puppet-bolt-x64-latest.msi.
2. Double-click the MSI file and run the installation.
3. Using Windows PowerShell, run a Bolt command to verify the installation.

```
bolt --help
```

If this doesn't work, check the following:

- Have at least PowerShell 3.0 or higher installed, as previous versions do not support PS Module auto-loading
- Run the following command to check why the module didn't load: `import-module PuppetBolt -Verbose`
- If the `import-module` command failed because unsigned scripts aren't allowed, please temporarily change the ExecutionPolicy to 'RemoteSigned' by running:

```
Set-ExecutionPolicy RemoteSigned
```

Windows Installation using Chocolatey

Chocolatey is an extremely simple to use package manager designed for Windows. For more information on Chocolatey, see chocolatey.org. You must have the Chocolatey package manager previously installed in order to use this installation method.

1. Using Windows PowerShell, download and install the bolt package:

   ```
   choco install puppet-bolt
   ```

2. run a Bolt command to verify the installation.

   ```
   bolt --help
   ```

If this doesn't work, check the following:

- Have at least PowerShell 3.0 or higher installed, as previous versions do not support PS Module auto-loading
- Run the following command to check why the module didn't load: `import-module PuppetBolt -Verbose`
- If the `import-module` command failed because unsigned scripts aren't allowed, please temporarily change the ExecutionPolicy to 'RemoteSigned' by running: `Set-ExecutionPolicy RemoteSigned`

MacOS using the installer

Use the Apple Disk Image (DMG) to install Bolt on macOS.

1. Download the Bolt installer package for your macOS version.
   Tip: To find the macOS version number on your Mac, go to the Apple menu in the top-left corner of your screen and choose About This Mac.
   - 10.11 (El Capitan): https://downloads.puppet.com/mac/puppet6/10.11/x86_64/puppet-bolt-latest.dmg
   - 10.12 (Sierra): https://downloads.puppet.com/mac/puppet6/10.12/x86_64/puppet-bolt-latest.dmg
   - 10.13 (High Sierra): https://downloads.puppet.com/mac/puppet6/10.13/x86_64/puppet-bolt-latest.dmg
   - 10.14 (Mojave): https://downloads.puppet.com/mac/puppet6/10.14/x86_64/puppet-bolt-latest.dmg

2. Double-click the **puppet-bolt-latest.dmg** file to mount it and then double-click the **puppet-bolt-[version]-installer.pkg** to run the installation.
3. From a terminal session, run a Bolt command to verify the installation.

   ```
   bolt --help
   ```

MacOS using Homebrew

Homebrew is a package manager designed for MacOS. For more information on Homebrew, see https://brew.sh. You must have the Homebrew previously installed in order to use this installation method.

3. Using a terminal session, download and install the bolt package:

   ```
   brew cask install puppetlabs/puppet/puppet-bolt
   ```

4. run a Bolt command to verify the installation.

   ```
   bolt --help
   ```

Lab 2: Running a simple command with Bolt

In this exercise we will run a simple “ping” command through bolt on a target host.

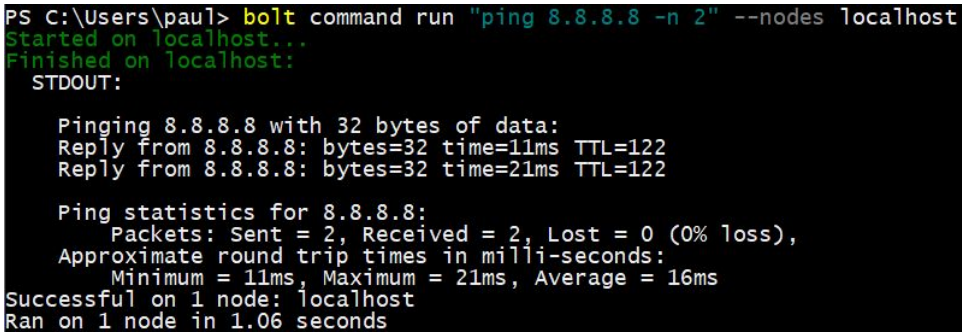
Since you will not have access to the remote lab test systems until the workshop, we will use “localhost” to target your local workstation. That means you can use Bolt to run things locally too! This is probably more useful when applying Tasks, Plans and PuppetCode, but it’s also sometimes useful for testing and will work as an example for usage during this lab. The process is slightly different between Windows and MacOS/Linux targets, so use the appropriate section for your workstation.

Windows:

1. Open a Windows PowerShell Prompt and run the following command:

   `bolt command run “ping 8.8.8.8 -n 2” --nodes localhost`

2. Observe the output of bolt. You should see something resembling the screenshot below:

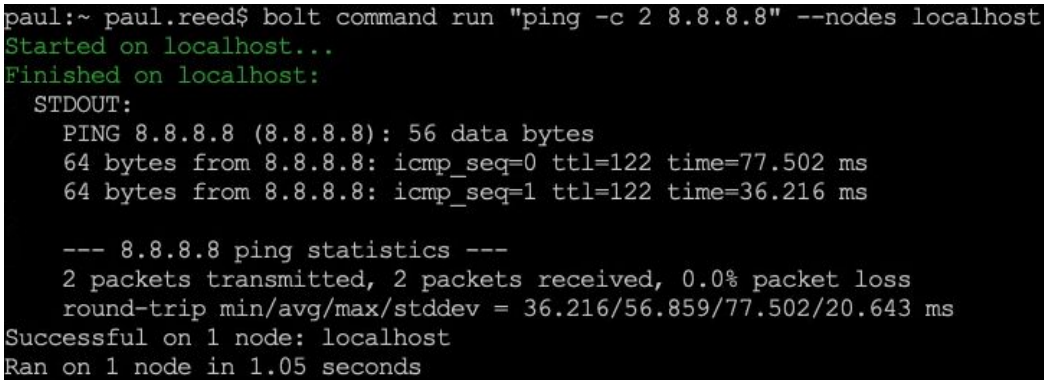


MacOS/Linux:

1. Open a Terminal Session and run the following command:

   `bolt command run “ping 8.8.8.8 -c 2” --nodes localhost`

2. Observe the output of bolt. You should see something resembling the screenshot below:



Note: “8.8.8.8” is the IP address of a Google DNS server which responds to ping requests. You can use any IP address or resolvable hostname you like to test with instead.

If you have access to a remote node with SSH or WinRM, you can try running commands against them as well. Make sure to use the “**winrm://**” prefix when targeting remote Windows nodes:

```
bolt command run ‘echo Hello World!’ --nodes winrm://<mynode> -u <user> -p
```

You can specify the “**ssh://**” prefix for SSH targets, but this is the default, so it is not required. Another possibility is to use the “**--transport**” option instead of a prefix:

```
bolt command run ‘echo Hello World!’ --nodes <node> --transport winrm -u <usr> -p
```

The “**-u**” option allows you to specify the user to connect on the target host.
The “**-p**” option allows you to specify the password. You don’t need to put the password on the command line, you will be asked to enter it once Bolt runs.

A Note on Interactive Processes and Bolt

It’s important to use the “-c 2” option with the “ping” command when running against a Mac or Linux target. If this option is omitted, the invoked command will run continuously and never return control back to Bolt. If you accidentally omit this option, press CTRL-C and bolt will interrupt the current run.

This is not an issue for the “ping” command on Windows targets. The ping command on Windows defaults to 4 pings, so it will return quickly if the option is omitted.

This exemplifies the need for invoked commands and scripts to run without user interaction with Bolt. A change of workflow using Tasks and Plans are likely a better option. These will be covered in the workshop.

Another option for interactive sessions with Bolt is to use what’s known as “expect scripts”, which could be written in PowerShell or Bash and used with a “bolt script run” command. We’ll cover using scripts with Bolt in the workshop as well.

Lab 3: Installing Microsoft Visual Studio Code

Microsoft Visual Studio Code (VS Code) is a lightweight, open source, multi-platform code editor. It is ideally suited for our use with Bolt in developing scripts and configuration files for multiple languages. It includes syntax highlighting, linting and integrates well with source control. It's also highly extensible via a multitude of extensions.

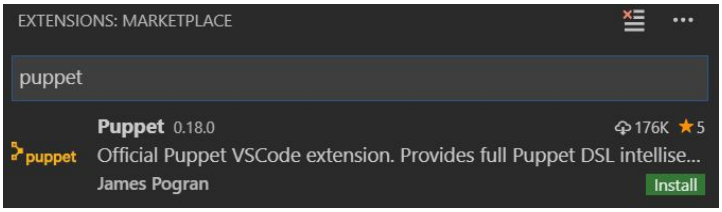
Downloads are available from https://code.visualstudio.com.

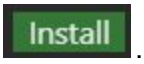
1. Browse to https://code.visualstudio.com and download the installer applicable to your OS, or use one of the following options:
    a. Windows Chocolatey users: "**`choco install vscode`**"
    b. MacOS Homebrew users: "**`brew cask install visual-studio-code`**"

Lab 4: Installing the Official Puppet VS Code Extension

To make things a little easier, we're going to get VS Code configured with the Official Puppet Extension. This will allow for syntax highlighting, intelliSense for resources, parameters, linting and more.

1. Open VS Code and click the extensions icon  to open the extension pane.
2. Use the search bar and search for "Puppet". The first item listed should be the Official Puppet VSCode extension by James Pogran.



3. Install the extension by pressing the green "Install" button  .

    Once installed, the button will be replaced with a "gear" icon  .