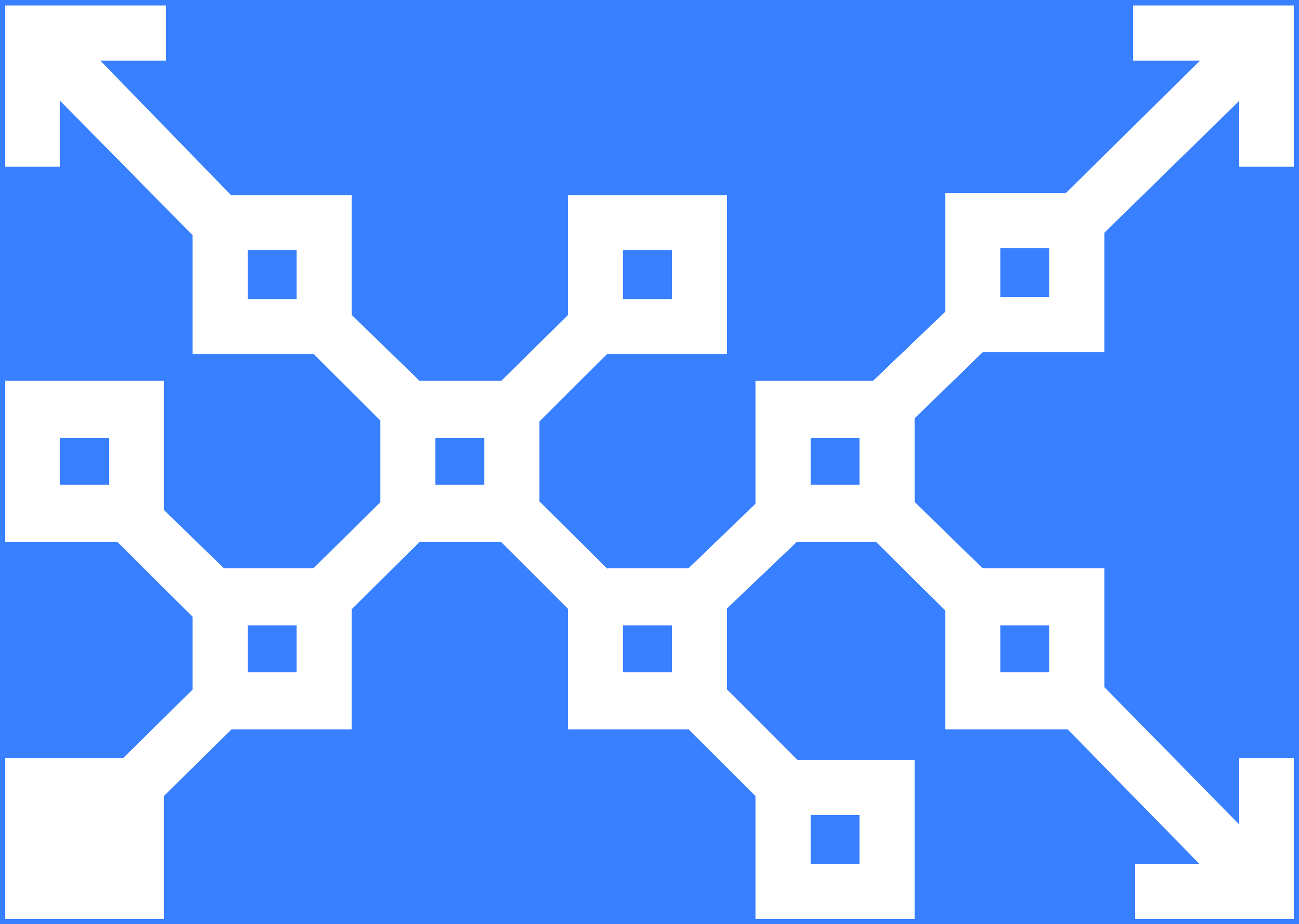
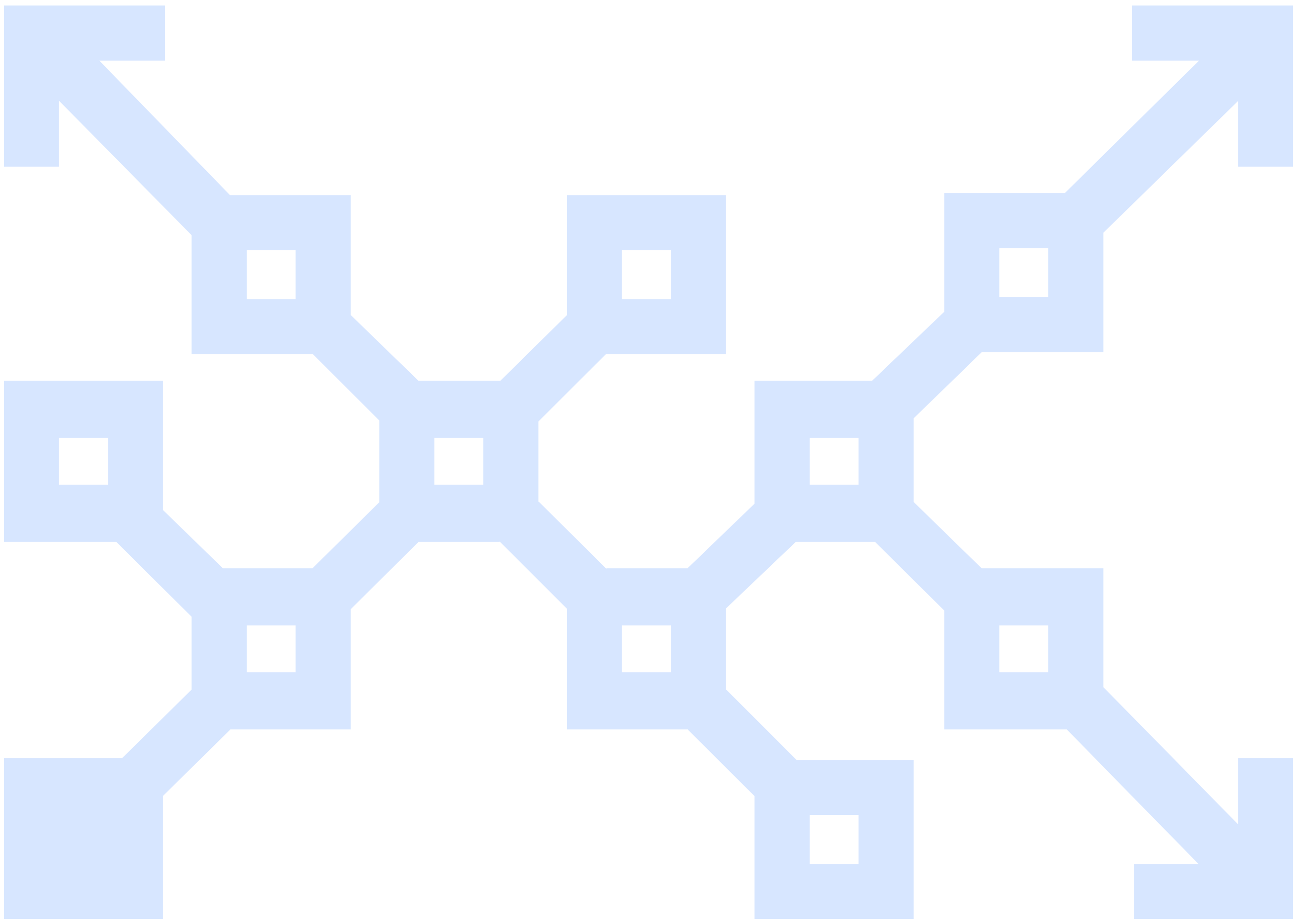
WHITE PAPER

Puppet Enterprise Test Drive on Azure



puppet

Contents



Welcome

Welcome to the Puppet Enterprise test drive on Azure!

In this 15-minute test drive, we'll dive in and see what Puppet can do. You will use the Puppet Enterprise (PE) console to set up Graphite, an open-source graphing tool that lets you easily visualize the state of your infrastructure. Graphite, like Puppet, spans the gap between nuts-and-bolts and the big-picture, which makes it a good example to get you started on your path to Puppet mastery.

While typically Puppet Enterprise is run in a Master-Agent architecture, in this example, we will simulate both the Puppet master and Puppet agent on the same VM.

This guide will be your companion as you make your way through this exercise.

Microsoft now provides a version of the Azure SDK that contains PowerShell cmdlets for provisioning, enabling, and disabling the Puppet extension handler on Windows virtual machines (VMs). Essentially, this provides a command-line interface that allows users to deploy Puppet Enterprise agents to Windows VMs in Azure.

What is Puppet?

Puppet is an open-source IT automation tool. The Puppet Domain Specific Language (DSL) is a Ruby-based coding language that provides a precise and adaptable way to describe a desired state for each machine in your infrastructure. Once you've described a desired state, Puppet does the work to bring your systems in line and keep them there.

The easy-to-read syntax of Puppet's DSL gives you an operating-system-independent language to specify which packages should be installed, what services you want running, which users accounts you need, how permissions are set, and just about any other detail of a system you might want to manage. If you're the DIY type or have unique needs, you can write the Puppet code to do all these things from scratch. But if you'd rather not re-invent the wheel, a wide variety of pre-made Puppet modules can help you get the setup you're looking for without pounding out the code yourself.

Why not just run a few shell commands or write a script? If you're comfortable with shell scripting and concerned with a few changes on a few machines, this may indeed be simpler. The appeal of Puppet is that it allows you to describe all the details of a configuration in a way that abstracts away from operating system specifics, then manage those configurations on as many machines as you like. It lets you control your whole infrastructure (think hundreds or thousands of nodes) in a way that is simpler to maintain, understand, and audit than a collection of complicated scripts.

Puppet Enterprise (PE) is a complete configuration management platform, with an optimized set of components proven to work well together. It combines a version of open source Puppet (including a preconfigured production-grade Puppet master stack), with MCollective, PuppetDB, Hiera, and more than 40 other open source projects that Puppet Labs has integrated, certified, performance-tuned, and security-hardened to make a complete solution for automating mission-critical enterprise infrastructure.

In addition to these integrated open source projects, PE has many of its own features, including a graphical web interface for analyzing reports and controlling your infrastructure, orchestration features to keep your applications running smoothly as you coordinate updates and maintenance, event inspection, role-based access control, certificate management.

Task 1 - Getting Started

After test drive provisioning is complete, login credentials are provided in the environment window as well as by e-mail. With the username, password, and IP address provided, SSH into the Virtual Machine from your terminal or SSH client.

Log into your SSH session.

```
ssh -l azureuser <ipaddress>
```

And supply the `<password>` provided when prompted.

Task 2 - Using the Puppet Module Tool

In this quest you will use the Puppet Enterprise (PE) console to set up Graphite, an open-source graphing tool that lets you easily visualize the state of your infrastructure. Graphite, like Puppet, spans the gap between nuts-and-bolts and the big-picture, which makes it a good example to get you started on your path to Puppet mastery.

Graphite is built from several components, including the Graphite Django webapp frontend, a storage application called Carbon, and Whisper, a lightweight database system. Each of these components has its own set of dependencies and requires its own installation, and configuration. You could probably get it up and running yourself if you set aside a little time to read through the documentation, but wouldn't it be nice if somebody had already done the work for you?

You're in luck! Puppet Labs operates a service called the Puppet Forge, which serves as a repository for Puppet ***modules***. A module nicely packages all the code and data Puppet needs to manage a given aspect in your infrastructure, which is especially helpful when you're dealing with a complex application like Graphite.

The `puppet module` tool lets you search for modules directly from the command line. See what you can find for Graphite. (If you're offline and run into an error, look for instructions below on installing a locally cached copy of the module.)

Run the following commands.

```
puppet module search graphite
```

Cool, it looks like there are several matches for Graphite. For this task, we will use Daniel Werdermann's module: `dwerder-graphite`.

Task 3 - Installing a Puppet Module

Now that you know what module you want, you'll need to install it to the Puppet master to make it available for your infrastructure. The `puppet module` tool makes this installation easy.

Go ahead and run:

```
puppet module install dwerder-graphite
```

Easy enough, but what did we do, exactly?

When you ran the `puppet module` command, Puppet retrieved the graphite module from Forge and placed it in the Puppet master's ***modulepath***. The modulepath is where Puppet will look to find Puppet classes and other files and resources made available by any modules you download or create. For Puppet Enterprise, the default modulepath is `/etc/puppetlabs/code/environments/production/modules`.

Class and classification

The `graphite` module includes Puppet code that defines a `graphite` ***class***. In Puppet, a class is simply a named block of Puppet code that defines a set of associated system resources. A class might install a package, customize an associated configuration file for that package, and start a service provided by that package. These are related and interdependent processes, so it makes sense to organize them into a single configurable unit: a class.

While a module can include many classes, it will generally have a main class that shares the name of the module. This main class will often handle the basic installation and configuration of the primary component the module is designed to manage.

The `graphite` class contains the instructions Puppet needs to set up Graphite, but you still need to tell Puppet where and how you want it to apply the class across your infrastructure. This process of matching classes to nodes is called ***classification***.

Task 4 - Log into the PE console

Now that you have the module installed, let's go through the steps of creating a node group, adding the Learning VM to the group, and classifying the group with the graphite class.

But before you can access the PE console you'll need the Learning VM's IP address.

Of course, you could use a command like `ifconfig` to find this, but let's do it the Puppet way. Puppet uses a tool called facter to collect facts about a system and make them available at catalog compilation. This is how it knows, for example, whether it's on Ubuntu and needs to use apt-get or CentOS and needs yum. You'll learn more about facts and conditionals in Puppet later. For now, we can use facter in the command-line to determine the Learning VM's IP address.

Run the following:

```
facter ipaddress
```

Open a web browser on your host machine and go to `https://<ipaddress>`, where `<ipaddress>` is the IP Address you just discovered. (Be sure to include the s in https!)

The PE console certificate is self-signed, so your browser may give you a security notice. Go ahead and bypass this notice to continue on to the console.

You'll now see a screen prompting you to supply credentials.

To get the password, run the following from your ssh session:

```
sudo cat /etc/puppetlabs/installer/database_info.install | grep something something
```

ow, with password in hand, log in to the console:

Username: admin

Password: <insert password>

Task 5 - Create a Node Group

Now that you have access to the PE console, we'll walk you through the steps to classify your VM node with the `graphite` class.

First, create a node group called "Azure VM". Node groups allow you to segment all the nodes in your infrastructure into separately configurable groups based on the node's certname and all information collected by the facter tool.

Click on Classification in the console navigation bar. It may take a moment to load.

From here, enter “Azure VM” as a new node group name and click Add group to create your new node group.

Click on the new group to set the rules for this group. You only want the Azure VM to be in this group, so instead of adding a rule, use the Pin node option to add the node individually.

Click on the Node name field, and you should see the Azure VM’s certname autofilled. If no matching certname appears, trigger a Puppet run (`puppet agent -t`) on the Azure VM. As part of the Puppet run, the Azure VM will check in, making its information available to the console node classifier.

Click Pin node, then click the Commit 1 change button in the bottom right of the console interface to commit your change.

Task 6 - Add a class

When you installed the `dwerder-graphite` module from the forge, it made the `graphite` class available in the console.

Under the ***Classes*** tab in the interface for the Azure VM node group, find the ***Class name*** text box. If `graphite` is not yet available, click the Refresh button near the top right of the classes interface and wait a moment before trying again.

Once you have entered `graphite` in the ***Class name*** text box, click the ***Add class*** button.

Before you apply the class, there are a few parameters you’ll want to set.

We already have an Apache server configured to our liking on the Azure VM, so we can tell the `graphite` class it doesn’t need to bother setting up its own server.

There are also some compatibility issues with the latest Django version. The author of this `graphite` module has made it easy to get around this problem by picking our own compatible Django version to use.

Set the parameters, as follows:

1. `gr_web_server = none`
2. `gr_django_pkg = django`
3. `gr_django_provider = pip`
4. `gr_django_ver = “1.5”`

Note that the `gr_django_ver` parameter takes a string, not float value, so it must be wrapped in quotes for Puppet to parse it correctly.

Double check that you have clicked the ***Add parameter*** button for all of your parameters, then click the ***Commit 5 changes*** button in the bottom right of the console window to commit your changes.

Task 7 - Run Puppet

Now that you have classified the Azure VM node with the graphite class, Puppet knows how the system should be configured, but it won't make any changes until a Puppet run occurs.

By default, the Puppet agent daemon runs in the background on all nodes you manage with Puppet. Every 30 minutes, the Puppet agent daemon requests a catalog from the Puppet master. The Puppet master parses all the classes applied to that node, builds the catalog to describe how the node is supposed to be configured, and returns this catalog to the node's Puppet agent. The agent then applies any changes necessary to bring the node into the line with the state described by the catalog.

To trigger a Puppet run, execute the following:

```
puppet agent --test
```

Graphite is a complex piece of software with many dependencies, so this may take a while to run. After a brief delay, you will see text scroll by in your terminal indicating that Puppet has made all the specified changes to the Azure VM.

Now that Graphite is up and running, its API is available for generating graphs suitable for including in a dashboard. Note that Graphite has only been running for a few minutes, so it may not yet have much data to chart. If you wait a minute and refresh the page in your browser, you will see the graph update with new data.

You can also check out the Graphite console running on port 90. (`http://<ipaddress>:90`)

Conclusion

Great job on completing the Puppet Enterprise test drive on Azure! You should now have a good idea of how to download existing modules from the Forge and use the PE console node classifier to apply them to a node. You also learned how to use the the facter command to retrieve system information, and the `puppet agent --test` command to manually trigger a Puppet run.