DevOps, as the name implies, is all about fostering a stronger partnership between development and operations disciplines. It’s a merger of people, process and technology aimed at driving greater efficiency and success in: bringing new apps and functionality to market quickly and in maintaining and updating these Apps throughout their lifespan.

**DEVOPS**

DevOps can help to ensure that operational requirements can be defined collaboratively upfront during the development process and followed throughout the lifecycle.

From the developer and test perspective, these infrastructure requirements just become part of the code – called infrastructure-as-code. This makes spinning up these IT compliant environments quick and easy.

Development and Operations become a virtuous motion as you plan, build, deploy and operate apps and gather continuous feedback that you integrate back into the development process.

As an IT Professional, there are parts of the DevOps lifecycle that you’ll play a bigger role in than others, but it should always be a partnership.

Together with your developers, you’ll want to

- Plan and define the development, test and production environments to ensure that the correct guardrails are in place to deliver supportable, production quality code.
- Right test criteria and SLAs for apps are in place before they are released into production.
• Placement of Apps meets your data sovereignty needs.

• Coordination with the development team is ongoing as is the monitoring and iteration of Apps in production.

INFRASTRUCTURE AS CODE

So how do you get a unified environment that makes this collaboration possible?

First, as you work with your developers, it’s worth knowing that Azure is an open platform with lots of choices to support the tools and frameworks that your developers are using – for example, if you wanted, you could run your whole Linux stack in Azure.

Now back to the concept of Infrastructure as Code which is a key principle of DevOps.

While your developer teams work to get new features out at a fast rate, this is the main way to ensure that you supply them with a consistent, compliant and repeatable dev, test, QA and production environment. This allows you to use templates and scripts to define the infrastructure powering each of these environments.

Azure Resource Manager templates, written in JSON, provide the code that allows you to define the required infrastructure. You can choose from hundreds of templates available in Azure and from GitHub to get started.

SETTING UP DEVTEST LABS

To control resource utilization early on as Apps are being developed, you can gate these resources and carve up a dedicated area for development and testing within your Azure subscription. You could do this with the Role-based Access Controls in Azure, but we also have DevTest labs to make this easier.

This enables you to quickly and easily spin-up a pre-production, testing and staging environment for your developers with the appropriate permissions and scope to control costs.

• Start by establishing a budget for the lab – as a lab owner you select configuration and policies and select cost trend to set a dollar amount and target cost thresholds for admin notification.

• Next you assign resources to the DevTest Lab.

• If you have existing Azure Resource Manager templates that define your production environment, you can easily reuse these to deploy instances of the environment but sandboxed for development and test.

• Otherwise you can set up a new development and testing environment from scratch.

• As a lab owner, you just select the range of VM sizes.

• You can also set the VM quota per user or per lab, and define the VM running hours through auto-shutdown and auto-start.

• In addition to the cost control policies, you can also control the VM images and templates people can use in the lab.

• You can select appropriate images for your VMs from the marketplace or you can build your own custom images.

• And from here you can also specify formulas which allow you to further customize the base image – so that every time you set up a VM it will have a common set of properties, this will include artifacts which comprise apps that you may want to use for testing or other tools that you’d like to install or download.

• From there you’ll want to grant access to the DevTest labs environment.

• Just like Azure in general, this uses Azure Active Directory for authentication, allowing you to grant access to specific users and groups within your organization and you can provide access to external users.

• If you are familiar with Role-based Access Controls (RBAC) to assign specific access privileges, in DevTest labs we simplify this even further with predefined roles that enable access only to DevTest resources.

• These safeguards not only provide an additional layer of security but help you to manage costs ensuring Azure usage from only approved developers.

• You can then monitor usage and overall spend in the lab and get real-time cost projections to see how well you are tracking against budget.

DEVELOPER PERSPECTIVE CI AND CD

Once you have helped define the operational requirements of the application and test environment, responsibility for coding and building the application moves primarily to your developer counterparts.

In DevOps, two terms that you will hear about are Continuous Integration (CI) and Continuous Delivery (CD). The idea here is that you build and test smaller code batches, deploying more frequently. The more frequently you deploy, the more you can experiment, and the more opportunity you have to change what you are doing if it’s not working or continue to persevere.

In Visual Studio Team Services, during Continuous Integration, the build and testing of code is automated every time a team member commits changes to version control. This is important because developers often work
Continuous Integration as part of DevOps is a critical area for you to partner with your developer counterparts, so that you can influence the right IT operations criteria to test against, such as making sure that the App can scale or that you have accounted for load balancing, in order to catch issues early. From there, the build can be automatically picked up by your chosen Continuous Delivery tool.

In Visual Studio Team Systems, Release Management (shipment of the app) is automated to ensure the right approvals at each stage. Of course, your developers can continue to use their preferred open source tools, such as Jenkins, Travis CI, and Spinnaker at various stages of CI/CD to achieve the same.

Once the App is in production, the next step in the DevOps lifecycle sequence is Monitoring:

- Now both developer and IT teams can detect and diagnose issues.
- By using Azure Application Insights, you have access to live performance telemetry that ties directly back to code making it easier to make fixes.
- You can also track incident response and service health through Azure Monitor.
- Ultimately you can get to a point when you detect an issue before you get that helpdesk call.

The overall advantage is that because we have a standard process, you can keep production fresh with the shortest path from code check-in to deployment. And what’s more, the resulting code now has operational awareness baked in.

Azure provides a unified environment for the DevOps process, helping you to foster a mutually beneficial partnership with your developer stakeholders as you develop and ship new Apps and services.

**CREATE A NEW VIRTUAL MACHINE**

Virtual machines are the main resource used in a DevTest Labs environment. Let’s create a virtual machine to see how quickly we can provision these development resources. Microsoft offers a number of base Windows and Linux virtual machine images. We’ll use a Visual Studio base image for this example. Note that within the Visual Studio base image, you also have several options for different versions of Visual Studio and Windows, depending on your lab requirements.

**LAB OPTIONS**

On the DevTest Lab blade there are additional options to view your virtual machines and any that are available to be claimed. Any VM you create in the lab yourself is automatically claimed by you and will also show up under My virtual machines. A claimable virtual machine is a VM that is available to any lab user with the necessary permissions. A lab admin can prepare VMs with specific base images and artifacts, and save them to a shared pool. A lab user can then claim an available VM from the pool when they need one with that specific configuration.

You can see an overview of all VMs in the lab or any data disks that are available for use. Any data disks added to VMs within the DevTest Lab can be detached and moved to other VMs within the lab, if desired.

**CONFIGURATION AND POLICIES**

Having an environment like this to quickly deploy development systems empowers developers with quick access to the tools they need, instead of having to download and configure everything on a system to get an application up and running. Visual Studio is one of those tools common to deployments in DevTest Labs. Let’s see how in just a few more minutes we can create a web application and deploy it to Azure.
CREATE A WEB APP

Now that we’ve seen how quickly you can create pre-loaded virtual machines to speed development, let’s take a look at how quickly you can deploy web apps to Azure using Visual Studio. Running Visual Studio for the first time after provisioning a DevTest Lab virtual machine can take a couple of minutes.

VIEW THE WEB APP

In the Azure portal, we can now see the new resources that were deployed. In our Resource group we can see a new App service that matches the name we saw in Visual Studio. On the Overview page we can see the URL for our web app in the Essentials section and browse directly to it. Within just a few minutes we have published a web application to Azure.

ADDITIONAL RESOURCES

Streamline continuous deployment with DevOps tools that help you get innovative applications into user’s hands faster. You can learn more with these useful resources:

AZURE LEARNING PATHS

- Node.js Developer on Azure
- .NET Developer on Azure

HANDS-ON LABS

- Self-paced Labs

MICROSOFT MECHANICS

- Operationalizing DevOps practices with Azure and Azure Stack
- Application Insights – Live telemetry across the app lifecycle