Azure DevTest Labs Prescriptive Adoption Guidance

Contents

Introduction

Enterprises are adopting the cloud rapidly, due to its benefits from an agility, flexibility and economics perspective. A usual first step into the cloud is migrating Development and Test workloads, though there are a common set of concerns when migrating these workloads including-

- Securing Development/Test Resources
- Clear separation between Dev/Test and Production
- Division of resources between IT, Application, Project Teams
- Managing Cost
- Providing self-service without compromising security

Intended Audience

This document is intended for IT planners, architects, and managers who are responsible for establishing and reviewing overall deployments and oversee operations practices. As a result, this document emphasizes the overall process and recommended design principles to promote a secure and stable development environment which ultimately will drive adoption of Azure DevTest Labs within an organization.

Getting Started

Proof of Concept

Once an organization has decided to explore DevTest Labs there are two general paths forward – Proof of Concept vs Scaled Deployment. A Scaled Deployment consists of weeks/months of review and planning with intent of deploying DevTest Labs to the entire Enterprise of hundreds or thousands of developers. Alternatively, a Proof of Concept is focused on a concentrated effort with a single team to establish organizational value. While it can be tempting to think a Scaled Deployment will rapidly increase value creation, the approach tends to fail more often than the alternative option. Therefore, it is recommended to start small, learn from the first team, repeat the same approach with two to three additional teams then plan for a Scaled Deployment based on the knowledge gained. For a successful Proof of Concept, it is recommended to pick one, two teams maximum, and identify their scenario (Dev Environment vs Test Environments), document their current use cases and deploy DevTest Labs.

DevTest Labs Scenarios

There are three primary scenarios for a DevTest Labs implementation.

Developer Desktops

Developers often have varying development machine requirements for different projects. With DevTest Labs developers can have access to on-demand VM's with the pre-configured VM's based on the scenario's they most commonly need. In this scenario, DevTest Labs provides the following benefits:

- Organizations can provide common development and testing environments ensuring consistency across teams
- Developers can quickly provision their development machines on demand
- Developers can provision resources in a self-service way without needing subscription-level permissions
- IT or Admins can pre-define the networking topology and developers can directly leverage it in a simple & intuitive way without requiring any special access
- Developers can easily customize their development machines as needed
- Administrators can control costs by ensuring that:
    - Developers cannot get more VM's than they need for development
    - VM's are shut down while not in use
    - Only allowing a subset of VM instance sizes for the specific workloads

Test Environments

Creating and managing Test Environments across an Enterprise can require a significant effort. With DevTest Labs Test Environments can be created, updated, duplicated with the click of a button allowing teams access to a fully configured environment when it's needed. In this scenario, DevTest Labs provides the following benefits:

- Testers can test the latest version of their application by quickly provisioning Windows and Linux environments leveraging reusable templates and artifacts.
- Testers can scale up their load testing by provisioning multiple test agents
- Admins can connect the DevTest Lab to VSTS to enable DevOps scenarios
- Administrators can control cost by ensuring that:
    - Testers cannot get more VM's than they need
    - VM's are shut down when not in use
    - Only allowing a subset of VM instance sizes for the specific workloads

Labs/Training/Hackathon/Prototyping

An Azure DevTest Lab acts as a great container for transient activities like workshops, hands on labs, training, or hackathons. The service allows you to create a lab where you can provide custom templates that each trainee can use to create identical and isolated environments for training. You can apply policies to ensure that training environments are available to each trainee only when they need them and contain enough resources - such as virtual machines - required for the training. Finally, you can easily share the lab with trainees, which they can access in one click. After the class has concluded, it's easy to delete the DevTest Lab and all the related resources.

Rationalize Scale Up

Prior to implementing DevTest Labs at Enterprise-scale there are several key decision points. Understanding these decision points at a high level will help an organization with design decisions in the future however they should not hold back an organization from starting a Proof of Concept. The top three areas for initial scale-up planning are listed below.

Networking and Security

Networking and Security are a cornerstone for all organizations and while an enterprise wide deployment will require a much deeper analysis as outlined in the document below there are a reduced number of requirements to successfully accomplish a Proof of Concept. A few key areas of focus include:

- **Azure Subscription –** In order to deploy DevTest Labs the organization must have access to an Azure Subscription with appropriate rights to create resources. There are a number of ways to gain access to Azure Subscriptions including through an Enterprise Agreement or Pay As You Go. More information can be found on gaining access to an Azure Subscription here.
- **Access to On-Premises Resources –** Some organizations will require their resources in Dev/Test Labs have access to on-premises resources thus a secure connection to Azure is needed. Therefore, it is important an organization has setup/configured either a VPN or Express Route connection prior to getting started. More information on setting these up can be found here.
- **Additional Security Requirements –** Other security requirements such as machine policies, access to public IP addresses, connecting to the internet are scenario's that may need to be reviewed prior to a Proof of Concept. If the above mentioned are areas of concern the section below provides additional information. In addition, your Microsoft Account Team can assist with targeted discussions around security.

Subscription Topology

Subscription Topology will be a critical design consideration when deploying DevTest Labs to the Enterprise. However, it is not required to solidify all decisions until after a Proof of Concept has been completed. When evaluating the number of subscriptions required for an Enterprise implementation there are two extremes – One Subscription vs Infinite Subscriptions. Next, we'll highlight the Pros of each approach.

One Subscription

Often the approach of One Subscription is not manageable in a large Enterprise however limiting the number of subscriptions provides the following benefits:

- **Forecasting** costs for Enterprise budgeting becomes much easier in a single subscription because all resources are in a single pool thus allowing for simpler decision making on when to exercise cost control measures at any given time in a billing cycle.
- **Manageability** of VM's, Artifacts, Formula's, network configuration, permissions, policies, etc is easier since all the updates are only required in one subscription as opposed to making updates across many subscriptions.
- **Networking** effort is greatly simplified in a single subscription for enterprises where on-premises connectivity is a requirement. Connecting virtual networks across subscriptions (hub-spoke model) is required with additional subscriptions which requires additional configuration, management, IP Address spaces, etc.
- **Team Collaboration** is easier when everyone is working in the same subscription – for example, it's easier to reassign a VM to a co-worker, share team resources, etc.

Subscription Per User

A separate subscription per user provides equal opportunities to the alternative spectrum. The benefits of having many subscriptions includes:

- **Azure Scaling Quotas** are not going to impede adoption. For example, as of this writing Azure allows 200 storage accounts per subscription. There are operational quotas for most services in Azure (many can be customized, some cannot). In this model of a subscription per user, it's highly unlikely that most quotas will be reached. More information on current Azure Scaling Quotas can be found here.
- **Chargebacks** to groups or individual developers becomes much easier allowing organizations to account for costs leveraging their current model.
- **Ownership & Permissions** of the DevTest Lab environments are very simple, always give the developer subscription-level permissions and they are 100% responsible for everything including the networking configuration, lab policies and VM management.

In the Enterprise, very likely there are enough constraints on the extremes of the spectrum, this requires a compromise resulting in an end state somewhere in the middle. As a Best Practice, the goal of an Organization should be to leverage the minimal amount of subscriptions as possible keeping in mind the forcing functions that will increase the total number of subscriptions. To reiterate, Subscription Topology will be critical for an Enterprise deployment of DevTest Labs but should not delay a Proof of Concept. There are additional details below under "Governance" on how to decide on subscription and DevTest lab granularity in the organization.

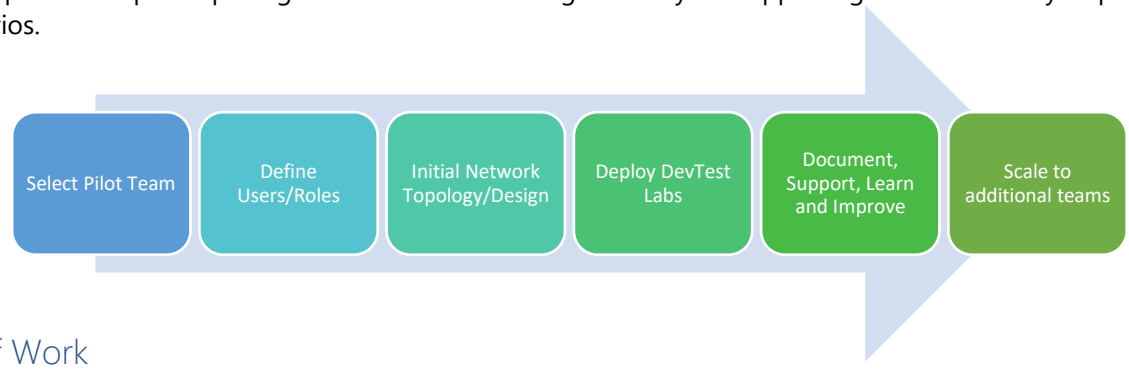
Roles and Responsibilities

A DevTest Labs Proof of Concept will have three primary roles with defined responsibilities – Subscription Owner, Owner, User and optionally a Contributor.

- **Subscription Owner –** The Subscription Owner will have rights to administer an Azure Subscription including assigning users, managing policies, creating & managing networking topology, requesting quota increases, etc. More information on a Subscription Owner can be found here.
- **DevTest Labs Owner –** The DevTest Labs Owner has full administrative access to the DevTest Lab. This person is responsible for add/removing users, managing cost settings, general lab settings and other VM/Artifact based tasks. A Lab Owner also has all the rights of a DevTest Labs User.
- **DevTest Labs User –** The DevTest Lab User will create and consume the Virtual Machines in the DevTest Lab. These individuals will also have some minimal administrative capabilities with VM's they create (start/stop/delete/configure their VMs). The users are unable to manage VMs of other users.

Orchestrating the Implementation of DevTest Labs

The following steps for implementing DevTest Labs are presented in a method and manner that provides enterprise clients, partners, and IT Pros a recommended approach for rapid deployment within Azure. The following emphasizes the overall process as prescriptive guidance while observing flexibility for supporting various industry requirements and scenarios.



Scope of Work

Assumptions

It is assumed the following items are in place prior to a DevTest Labs pilot:

- **Azure Subscription**: The pilot team has access to deploying resources into an Azure Subscription. If the workloads will be only development and testing, it's recommended to select the Enterprise DevTest offer for additional available images and lower rates on Windows virtual machines.
- **On-Premises Access:** If required, On-Premises access has already been configured. This can be accomplished via a Site-to-site VPN connection or via Express Route. Connectivity via Express Route can typically take many weeks to establish, it's recommended to have this in place before starting the project.
- **Pilot Teams**: The initial development project team(s) that will use DevTest Labs has been identified along with applicable development or testing activities and establish requirements/goals/objectives for those teams.

Milestone 1: Establish Initial Network Topology and Design

The first area of focus when deploying an Azure DevTest Labs solution is to establish the planned connectivity for the virtual machines. The steps below outline the necessary procedures:

1. Define **initial IP Address ranges** that will be assigned to the DevTest Lab Subscription in Azure. This requires forecasting the expected usage in number of VMs so you can provide a large enough block for future expansion.
2. Identify **methods of desired access** into the DevTest Labs (e.g. external / internal access). A key point in this step is to determine if Virtual Machines will have Public IP addresses (IE: accessible from the internet directly).
3. Identify and establish methods of connectivity with the rest of the Azure cloud environment and on-premises. If forced routing with Express Route is enabled, it's likely that the Virtual Machines will need appropriate proxy configurations to traverse the corporate firewall.
4. If VMs are to be **domain joined** determine whether they will join a cloud-based domain (AAD Directory Services for example) or an on-premises domain. For on-premises, determine which OU within Active Directory that the Virtual Machines will join and confirm users have access to join (or establish a service account that has the ability to create machine records in the domain)

Milestone 2: Deploy the Pilot DevTest lab

Once the network topology is in place, the first/pilot DevTest Lab can be created following the steps below.

1. Create initial DevTest Lab environment (Step by Step instructions can be found here)
2. Determine allowable VM images and sizes for use with the DevTest Lab. Decide if custom images will be uploaded into Azure for use with DevTest Labs.
3. Secure access to the lab by creating initial Role Base Access Controls (RBAC) for the DevTest Lab (Lab Owners and Lab Users). **Recommended** use synchronized Active Directory accounts with Azure Active Directory for identity with DevTest Labs.
4. Configure DevTest Labs to use policies such as schedules, cost management, claimable VMs, custom images or formulas.
5. Establish an online repository such as VSTS/Git.
6. Decide on the use of Public or Private Repos or combination of using both. Organize JSON Templates for deployments and long term sustainment.
7. **Optional:** Create custom artifacts if the need arises.

Milestone 3: Documentation, Support, Learn and Improve

The initial pilot teams may require in-depth support getting started. Leverage their experiences to ensure the right documentation and support is in place for continued rollout of Azure DevTest Labs.

1. Introduce the Pilot teams to their new DevTest Lab resources (demos, documentation)
2. Based on pilot teams experiences, plan & deliver documentation as needed
3. Formalize process for onboarding new teams (creating & configuring labs, providing access, etc)
4. Based on initial uptake, verify original forecasts of IP Address space is still reasonable & accurate
5. Ensure appropriate compliance and security reviews have been completed

Governance

Resources

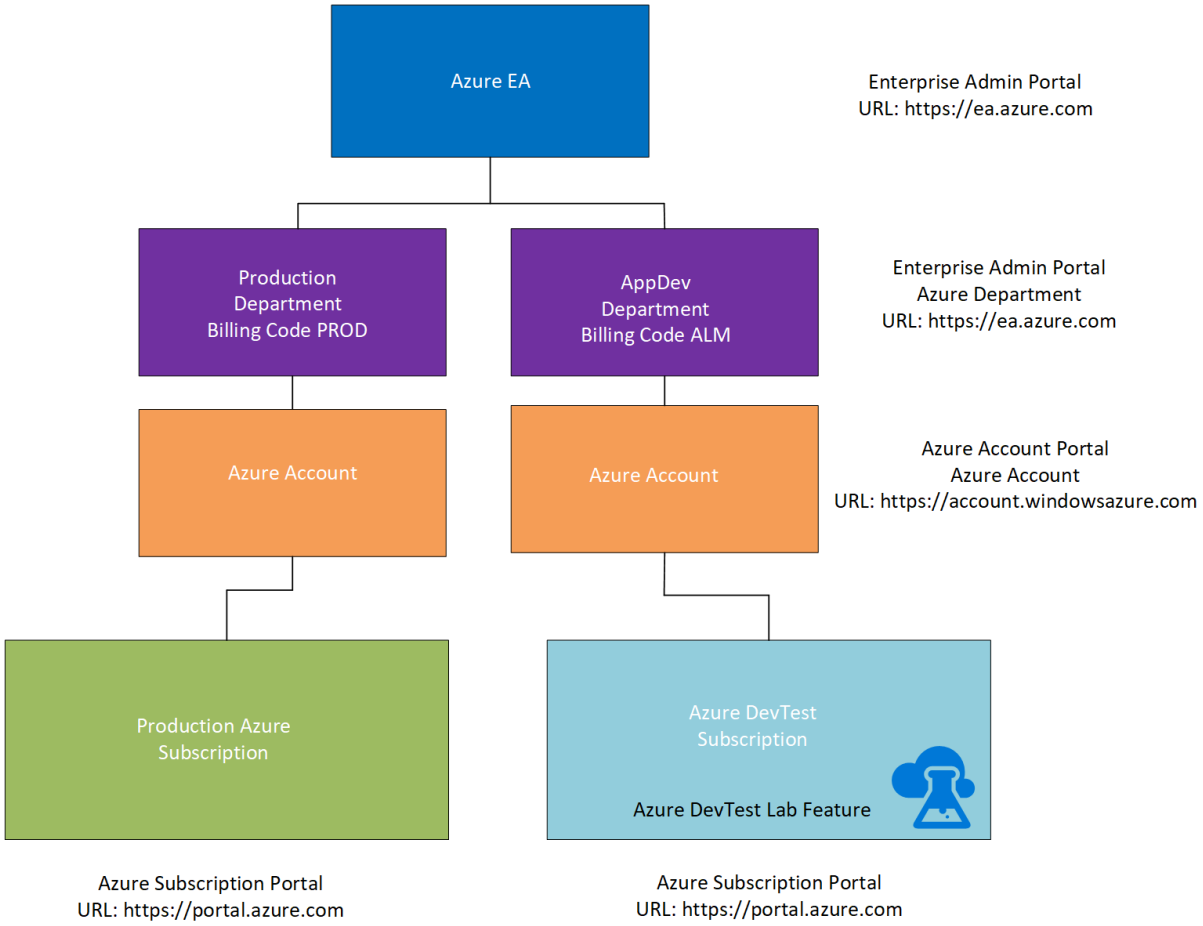
The following section addresses the alignment and management of resources for Azure DevTest Labs within an enterprise or a small to midsize business.

Q: How should DevTest Labs resources be aligned within an Azure subscription?

Before an organization begins to leverage Azure for general application development IT planners should first review how they will introduce the capability as part of their overall portfolio of services. Areas for review should address financial management to truly measure the cost associated with their application development lifecycle, how will the organization align the proposed service offering with their corporate security policy? Will segmentation be required between the Azure development environment and the production environment? The long-term business objectives should address the areas raised and what controls will be introduced for long term ease of management, stability, and growth.

The first recommended practice should be a review of the organizations Azure taxonomy where the division between the production and the development subscriptions are outlined. In the diagram below the suggested taxonomy would allow for a logical separation and the ability for the organization to introduce billing codes to support the business ability to track costs associated with each environment. There is additional documentation on this topic here. Additionally, Azure Tags can be introduced to support the organization of resources for tracking and billing purposes. Information about Azure Tags can be found here.

The second recommendation for an organization to optimize the most of their application development experience and cost management is to enable the DevTest Subscription within their Azure Enterprise Portal. This will allow an organization to run client operating systems that are not typically available in an Azure Enterprise Subscription and leverage enterprise software where they would pay just for the compute and not have to account for the licensing. This will ensure the billing for the designated services, including gallery images in IaaS such as Microsoft SQL Server will be consumption only. Details about the Azure DevTest subscription can be found here for Enterprise Agreement customers and here for Pay as you Go customers.



Another consideration for implementing this model provides a large organization the flexibility to deploy the Azure DevTest Labs at scale. An organization can support hundreds of labs for various business units that might have anywhere between one hundred to a thousand virtual machines running in parallel. This model promotes the notion of a centralized enterprise lab solution that can share in the same principles of configuration management and security controls.

This model also ensures that the organization will not exhaust their resource limits associated with their Azure subscription. As a reference please consult the following article regarding subscription service limits for planning services found here. An example of resource limits being consumption would be the Azure Resource Groups which currently have a service limit of 800 per subscription. The nature of the DevTest Labs provisioning process can consume large amounts of Resource Groups. Resources can be increased over time through a support request in the Azure DevTest subscription and the resources within the production subscription are not affected as the development subscription grows in use. The following article provides additional guidance on scaling DevTest Labs here.

A common subscription limit that needs to be accounted for when introducing Azure is how the network IP range assignments will be allocated to support both a production and development subscription to account for growth over time (assuming on-premises connectivity or another networking topology requiring the enterprise to manage their networking stack instead of defaulting to Azure's implementation). The recommended practice is to have few virtual networks that have a very large IP address prefix assigned and divided with many large subnets rather than to have multiple virtual networks with small subnets. For example, with 10 subscriptions we can define 10 virtual networks (one for each subscription) and all DevTest Labs that don't require isolation can share the same subnet on the subscription's vnet.

Q: **How to maintain a naming convention across my DevTest Labs environment?**

Another area for an organization to apply proper governance in Azure and DevTest Lab is to possibly extend its current enterprise naming convention to support Azure operations and make it consistent across the DevTest Lab environment.

When deploying DevTest Labs it is recommended to have specific starting policies which can be deployed by a central script and JSON templates to enforce consistency. Naming policies can be implemented through Azure policies applied at the subscription level. Implementation details can be found here.

Q: How to determine the ratio of users per DevTest Lab and the overall number of DevTest Lab needed across an organization?

As a recommended practice business units and development groups that are associated in the same development project should be associated within the same DevTest Lab. This will allow for scales of economy where the same types of policies, images, and shutdown policies can be applied.

Another boundary that an organization may want to consider are geographic. Developers in the North East United States may leverage a DevTest Lab provisioned in East US2, while developers in Dallas, Texas and Denver, Colorado may be directed to use a resource in South Central. If there is a collaborative effort with an external third party, they could be assigned to a lab where they are separated from the primary resources that have been assigned for the organization and internal developers if policy dictates this.

Another option recommended practice for consideration may be to leverage a project per lab with a specific project within Visual Studio Team Services team project and further apply security trimming through a specified Azure Active Directory Group which allow access to both set of resources. The virtual network assigned to the lab can be another boundary to consolidate users.

Q: How can we prevent the deletion of resources within a Dev Test Lab?

The recommended practice to avoid deletion or changing of the policies applied at the DevTest Lab is to apply proper permissions. All developers should be placed within the DevTest Lab Users group except the lead developer or infrastructure lead who should be the DevTest Lab Owner. A recommended practice is to only have two DevTest Lab Owners when the groups are large in number. This policy would also extend towards the code repository to avoid corruption. DevTest Lab Users have rights to use resources but cannot affect changes that have been defined with DevTest Lab Policies. Please refer to the following article that lists the roles and rights each built in group has within a DevTest lab which is documented here.

Q: Is it currently supported to move a DevTest Lab into another Resource Group?

Currently it is not supported to move a DevTest Lab from one resource group into another. If the activity is influenced by trying to align the DevTest Lab to an enterprise naming convention set for resource groups rather than accept the auto generated naming convention you have two options. You could either deploy the resource from PowerShell into an existing resource group, or open a pre-created resource group and then click "+ Add" to add a DevTest Lab into that resource group. In the future releases of DevTest Labs Microsoft may provide the ability for a stricter naming convention as resource groups are currently auto generated during deployments.

Managing Cost & Ownership

Cost and ownership are primary concerns, when you consider building out your Development and Test environments. In this section, you will find information to help you optimise for cost and align ownership across your environment that makes sense for you.

Q: How can I optimise for cost within my DevTest labs environment?
There are a number of built-in features as part of DevTest labs to help you optimise for cost, including Cost Management, Thresholds and Policies to limit the activities of your users.

As you are utilising DevTest labs for a DevTest workload, you may consider utilising the Enterprise Dev/Test Subscription Benefit, as part of your Enterprise Agreement. Alternatively, if you are a Pay as you Go customer, you may want to consider the Pay-as-you go DevTest offer.

This will provide you with numerous advantages;

- Special lower Dev/Test rates on Windows Virtual Machines, Cloud Services, HDInsight, App Service, and Logic Apps
- Same great EA rates on other Azure services
- Access to exclusive Dev/Test images in the Gallery, including Windows 8.1 and Windows 10

Only active Visual Studio subscribers (standard subscriptions, annual cloud subscriptions and monthly cloud subscriptions) can use the Azure resources running within an Enterprise Dev/Test subscription, though end users can also access the application to provide feedback or perform acceptance tests. Use of resources within this subscription is restricted to developing and testing applications, and no uptime guarantee is offered.

If you decide to use the DevTest offer, be aware that This benefit is exclusively for development and testing your applications. Usage within the subscription does not carry a financially-backed SLA, except for use of Visual Studio Team Services and HockeyApp.

Q: How do I define a pattern across my organization for RBAC in my DevTest Labs Environments to ensure IT can govern while Developers and Testers can continue their work?

There is a broad pattern, however the detail will depend on your organization.

Central IT should own only what is necessary, enabling the project and application teams to remain with the needed level of control. Typically, this would mean that central IT would own the subscription and handle core IT functions such as networking configurations. The set of owners for a subscription should remain small. Additionally, these owners can then nominate additional owners if needed, or apply subscription-level policies, e.g. “No Public IP”.

There may be a subset of users that require access across a subscription, such as Tier1 or Tier 2 support. In this case, it may be recommended to give these users Contributor access, meaning that they can manage the resources, but not provide user access or adjust policies.

The DevTest labs resource would then be owned as close to the project/application team as possible, as they will understand their requirements in terms of machines, and required software. In most organizations, the owner of this DevTest labs resource would commonly be the project or development lead. This owner can manage the users and policies within the Lab environment and can manage all VMs in the DevTest labs environment.

The DevTest labs user role would be provided to the Project / Application team. These users can Create Virtual Machines (in line with the lab and subscription-level policies) but can also manage their own Virtual Machines. They are unable to manage those Virtual Machines that belong to other users.

If you have not already done so, it is worth familiarizing yourself with the Azure enterprise scaffold – prescriptive subscription governance documentation.

Company Policy & Compliance

Q: When should an organization use the public artifact repository in DevTest Labs and when should a new private artifact repository be created?

The public artifact repository that is provided with DevTest Labs provide a rich source of content to help organizations have an initial common set of software packages for rapid deployment and consumption without having to invest time to reproduce common developer tools and "add ins". An organization can choose to deploy their own private repository. A public and private repository can be leveraged in parallel or an organization may choose to disable the public repository. The criteria to deploy a private repository should be driven by the following questions and considerations:

- Will the organization have a requirement to have corporate licensed software as part of their DevTest Lab offering? If the answer is yes, then a private repository should be created.
- Is the organization developing custom software as part of their application development that provides a specific operation and is required as part of the overall provisioning process? If the answer is yes, then a private repository should be deployed.
- If the organization has developed a governance policy where isolation is required, and external repositories are not under direct configuration management by the organization then a private artefact repository should be deployed. As part of this process an initial copy of the public repository can be copied and integrated with the private repository and the public repository can be disabled so that no one within the organization can access it anymore. This will force all users within the organization to have only a single repository that is approved by the organization and minimize configuration drift.

Q: Should an organization plan for a single repository or allow multiple repositories?

As part of an organizations overall governance and configuration management strategy the elimination of multiple repositories that may become silos of unmanaged software overtime is advisable as a recommended practice. A central repository where multiple teams may be able to consume for their projects is highly advisable to enforce standardization, security, ease of management, and eliminates the duplication of efforts. As part of the centralization the following are recommended practices for long term management and sustainability:

- Associate the Visual Studio Team Services with the same Azure Active Directory Tennant that the Azure subscription is leveraging for authentication and authorization.
- Create a "*All DevTest Lab Developers*" group in Azure Active Directory that is centrally managed. Any developer that is contributing to artefact development should be placed within this group.
- The very same "*All DevTest Lab Developers*" Azure Active Directory group can be leveraged to provide access to the Visual Studio Team Services repository and to the DevTest Lab.
- In Visual Studio Team Services branching or forking should be leveraged to a separate "in development" repository from the primary production repository. Content is only added to the master branch with a pull request after proper code review. Once the code review has provided an approval process should the updated content merged with the master branch by a lead developer tasked with the maintenance of the master branch.

Q: How can an organization ensure corporate security policies are in place, if needed with Active Directory?

An organization may achieve this by first developing and publishing a comprehensive security policy where it articulates the rules of acceptable use associated with the using software, cloud assets, and what clearly violates the policy. The second action that an organization can do is to develop a custom image, custom artifacts, and a deployment process that allows for orchestration within the security realm that is defined with active directory. This enforces the corporate boundary and sets a common set of environmental controls. These controls against the environment a developer can consider as they develop and follow a Secure Development Lifecycle (SDL) as part of their overall process. The objective also should provide an environment that is not overly restrictive that may hinder development, but a reasonable set of controls . The group policies that are targeted against the organization unit (OU) which will contain the DevTest Lab virtual machines systems could be a subset of the total group policies that are found in production or an additional set to properly mitigate any identified risks.

Q: How can an organization ensure data integrity (DLP) if a developer is remoting in and ensure that they are not removing code (IP) out or introducing malware or unapproved software?

There are several layers of control on how this can be addressed to mitigate the threat from external consultants, contractors or employees that are remoting in to collaborate in DevTest Labs. As stated previously the first step must have an acceptable use policy drafted and defined that clearly outlines the consequences if someone violates the policy. Although controls can be implemented to restrict theft within an organization. The first layer of controls for remote access to is to apply a remote access policy through a VPN connection that is not directly connected to the DevTest Lab. The second set of controls would be to apply a set of group policy objects that prevent copy and paste through remote desktop. A network policy could be implemented to not allow outbound services from the environment such as FTP and RDP services out of the environment. User defined routing could force all Azure network traffic back to on-premises, but this could not account for all URL's that might allow uploading of data unless controlled through a proxy to scan content and sessions. Public IP's could be restricted within the Virtual Network supporting the DevTest Lab to not allow bridging of an external network resource.

Ultimately the same type of restrictions would have to be applied across the organization which would have to also account for all possible methods of removable media or external URL's that could accept a post of content. Please consult with your security professional to review and implement a security policy. Also consult the following site for follow up recommendations: Microsoft Cyber Security.

Application Migration / Integration

Once your Development and Test lab environment has been established, how do you then begin utilizing that environment within your project team? How do you ensure that you are following any required organizational policies, and maintaining the agility to add value to your application?

Q: When should I use the Azure Marketplace image vs. bring my own custom organizational image?
Azure Marketplace should be used by default unless you have specific concerns or organizational requirements. Some common examples include;

- Complex software setup requires application to be included as part of the base image
- Installation and setup of application could take many hours, and is not an efficient use of compute time to be added on top of an Azure Marketplace image
- Developers and Testers require access to a Virtual Machine quickly, and want to minimize the setup time of a new Virtual Machine
- Compliance or regulatory conditions (for example, security policies) that must be in place for all machines.

Using custom images should not be considered lightly. This introduces extra complexity, as you will now have to manage VHD files for those underlying base images and will need to routinely patch those base images with software updates. This will apply for new Operating System updates, but also any updates or configuration changes needed for the software package itself.

Q: When should I use a formula vs. custom image?
Typically, the deciding factor in this scenario is cost and reuse.

If you have a scenario where many users/labs require a base image that installs a lot of software on top of that base image, then this cost could be saved by creating a custom image instead. This would then mean that the image is created once, and reduces the setup time of the Virtual Machine and the cost incurred due to the Virtual Machine running when setup occurs.

However, an additional factor to note is the frequency of your software package changes. If you are running daily builds and require that software to be on your users' virtual machines, then you should consider using a formula instead of a custom image.

Q: How can I setup an easily repeatable process to bring my custom organizational images into a DevTest labs environment?

The Visual Studio Customer Engagement team have documented a pattern called the Image Factory pattern. You can find more information about this pattern on channel 9.

This is an advanced scenario, and the scripts provided are sample scripts only. If any changes are required, you will need to manage and maintain the scripts used in your environment.

Using DevTest labs to create a custom image pipeline in Visual Studio Team Services (VSTS)

- Introduction: Get VMs ready in minutes by setting up an image factory in Azure DevTest Labs
- Image Factory – Part 2! Setup VSTS and Factory Lab to Create VMs
- Image Factory – Part 3: Save Custom Images and Distribute to Multiple Labs
- Video: Custom Image Factory with Azure DevTest Labs

Q: I may have a scenario where I need to ensure Development and Test Virtual Machines are unable to reach the public internet. Are there any recommended patterns to set up network configuration?
Yes. There are two aspects to consider – Inbound and Outbound Traffic.

Inbound Traffic – If the Virtual Machine does not have a Public IP Address, then it cannot be reached by the internet. A common approach is to ensure that a subscription-level policy is set, such that no user is able to create a public IP address.

Outbound Traffic – If you want to prevent Virtual Machines going directly to public internet and force traffic through a corporate firewall, then you can route traffic on premises via express route or VPN, by using Forced Routing.

Note: Do not forget that you will need to add exceptions to the Lab's Artefact storage account, if you have a proxy server that will block traffic without proxy settings.

You could also use Network Security Groups for Virtual Machines or subnets. This is an additional layer of protection to allow / block traffic.

Q: When should I create a new Virtual Network for my DevTest labs environment vs. using an existing Virtual Network?
If your Virtual Machines need to interact with existing infrastructure, then you should consider using an existing Virtual Network inside of your DevTest labs environment. Additionally, if you are using ExpressRoute, then you may want to minimise the amount of VNets / Subnets – this will help insure you don't fragment your IP address space that gets assigned for use in the subscriptions. You should also consider using the VNet peering pattern here (Hub-Spoke model). This enables vnet/subnet communication across subscriptions within a given region although peering across regions is an up-coming feature in Azure networking.

Otherwise, each DevTest lab environment could have its own Virtual Network. However, do be aware that there are limits on the number of Virtual Networks per subscription. The default amount is 50, though this can be raised to 100.

Q: When should I use a Shared IP, Public IP or Private IP?
If you are using a Site to Site VPN or Express Route, then you would likely consider using Private IPs, so that your machines are accessible via your internal network, and inaccessible over public internet. NOTE: Lab Owners can change this subnet policy, to really ensure that no one will accidently create public IP addresses for their VMs the subscription owner should create a subscription policy preventing public IPs from being created.

When using a shared public IPs, the virtual machines in a lab share a public IP address. This can be helpful when you need to avoid breaching the limits on Public IP addresses for a given subscription.

Q: Is there a rule in terms of how many Virtual Machines I should set per user, or per lab?
When considering the number of Virtual machines per user or per lab, there are three main concerns:

- The **overall cost** that the team can spend on resources in the lab. It's quite easy to spin up many machines, to control costs, one mechanism is to limit the number of VMs per user and/or per lab
- The total number of virtual machines in a lab is impacted by the **subscription level quotas** available. One of the first upper limits is 800 resource groups per subscription, and DevTest Labs currently creates a new resource group for each VM (unless shared public IPs are used). If there are 10 DevTest Labs in a subscription, that would mean labs could fit approx. 79 virtual machines in each lab. (800 upper limit – 10 RGs for the Labs / 10 Labs) = 79 Virtual machines per lab.
- If the lab is connected to on-premises via Express Route (for example), there will be a **defined IP address space available** for the Vnet/Subnet. To ensure that the VMs in the lab won't fail to be created (error: can't get IP address), lab owners can specify the max VMs per lab aligned with the IP address space available.

Q: How can I use ARM templates inside of my DevTest Labs Environment?

You are able to deploy your ARM templates into a DevTest labs environment by using the Environments feature in DevTest labs. To achieve this, you should check your ARM Templates into a Git Repository (either Visual Studio Team Services or GitHub), and add a private repository for your templates.

This scenario would not be useful if you are using DevTest labs to host development machines, but may be useful if you are building out a staging environment which is representative of production.

It is also worth noting that the "Limit number of Virtual Machines" per lab or per user option only limits the number of machines natively created in the DevTest Lab itself, and not any environments (ARM Templates).