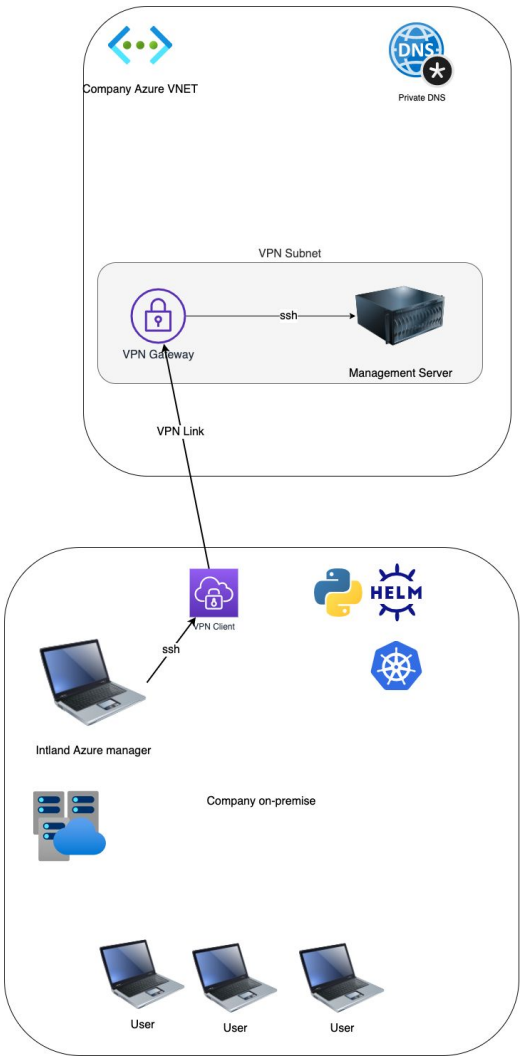
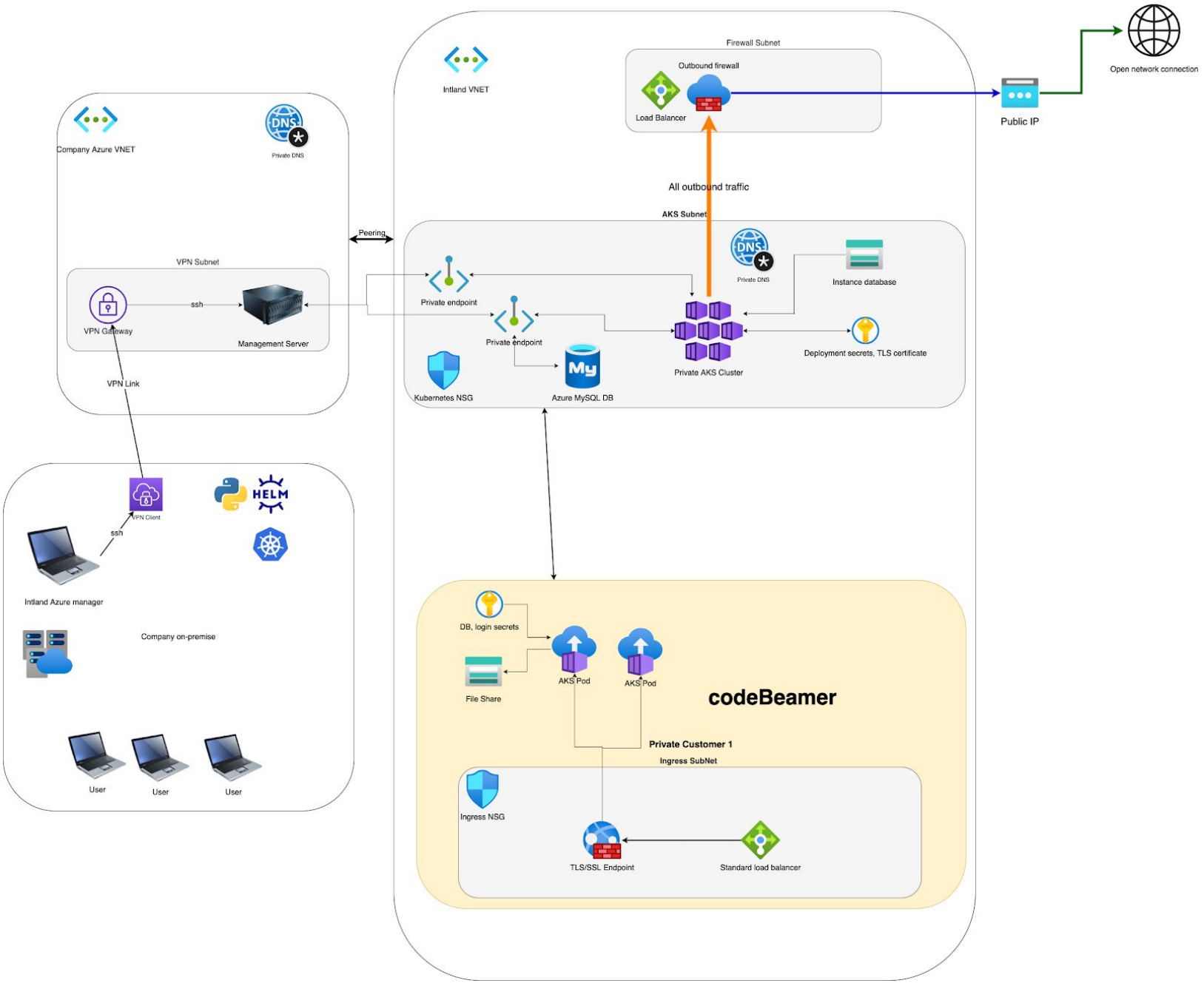
codeBeamer on Azure - v1.1

Auto-deployment overview

The below described system automatically creates a codeBeamer environment applying all security requirements mentioned in the next section.



The base configuration, which needs to exist before the deployment is shown on the diagram above. After the deployment is completed the architecture looks like this:



Security

The deployment is created in a new virtual network, which is linked to your existing virtual network in Azure. The most common scenario is that the codeBeamer is running in a separate virtual network to which a peering is enabled, so it can also reach services in your virtual network. All resources (storage, key vault, MySQL server, Kubernetes cluster) are located in the automatically created virtual network.

Public IP policy: there are no inbound public IPs enabled in the deployment process. This means the service is not accessible from the open network in any way. The deployment uses private endpoints and a private Kubernetes cluster. There is only a single public IP for outbound traffic, for which a Firewall service can be optionally enabled, so the complete traffic can be monitored to the service.
All created DNS zones are private and only available for the deployment and kubernetes subnetworks.

Deployment

In order to create and access this deployment there needs to be a virtual machine called jumpbox, which is located in the virtual network, from which you will enable access to the newly created cluster. This is management access, the users will not be able to reach this service from the application endpoint.
The deployment location is recommended to be “Germany west central” if you are located in Germany, as this location has all services available.

Infrastructure requirements

1. Select a DNS zone and deployment name
    a. DNS zone can be: customer.com
    b. Deployment name can be: alm, codebeamer, any
    c. The FQDN will be **codebeamer.customer.com**
2. Obtain a TLS certificate for the above mentioned FQDN. This can be a wildcard certificate for the DNS zone or a specific for the deployment name. Wildcard certificates are preferred. Store the **.cert** and **.key** files locally, they are needed for the deployment.
3. You need to have at least “Network Contributor” access to the vnet to which the newly created vnet will be linked.
4. Your subscription resource providers needs to be enabled for the following: *Microsoft.Compute, Microsoft.Network, Microsoft.KeyVault, Microsoft.DBforMySQL, Microsoft.Storage, Microsoft.ContainerService*
5. If you want to enable email from the service the recommended way is using Sendgrid. Please deploy a Sendgrid account to enable the notification service from codeBeamer. For this the *Sendgrid.Email* resource provider needs to be enabled. Account can be set up for use here: https://sendgrid.com/docs/for-developers/partners/microsoft-azure/. Using Sendgrid is not required, any custom email server can be used if it is accessible from your virtual network.
6. Deploy a virtual machine into the “HUB” virtual network, this will be used as a jumphost to manage the cluster. We recommend using a standard Ubuntu machine with Docker installed (can be any Linux or Mac OS version). Do not enable any public IP addresses for the machine and explicitly ban SSH access from the open network for security reasons.

7. The above machine is not required if your machine is already in the virtual network and private DNS lookups work. This is not possible to do with point to site VPN on Azure. If you want to run the deployment on your local machine, I recommend using sshuttle which is a VPN over SSH connection and enables you to proxy the dns lookups too. Private DNS access can be easily checked by creating a zone and trying to nslookup it from the terminal.
8. Resource Quotas: E4s_v3 - 4 cores B2s - 6 cores are required for the recommended first setup, which is part of the default quota. To avoid problems please crosscheck it. In the deployment location you need to have premium file shares available.
9. You need to have Azure DB for MySQL available in your subscription in the needed zone, it can be checked on the Azure Portal if you can select the given location for the service.
10. The deployment script is supplied as a zip, and the environment is also given in a Docker image, so the setup is simple.
11. Run `az login` in the terminal and login to the subscription with your user.
12. Create an SDK service principal, which will be used by the script as follows:

```
export INTLAND_AZURE_AUTH_LOCATION=${HOME}/.azure/INTLAND_azure_credentials.json
az ad sp create-for-rbac --sdk-auth --name IntlandPrincipal > $INTLAND_AZURE_AUTH_LOCATION
```

Requirements

- Azure CLI installed and logged in with `az login` on server
- Docker and docker-compose installed, privileges set up for user
- Machine is either in the VNET or VPN connection to it (with remote DNS)
- Deployment VNET is not peering another VNET with the same subnet IP CIDR
- No other Kubernetes or MySQL private DNS is linked to the deployment VNET
- Certificates available below the home folder

Deploy service

Set environment variables as follows (can be added to .zshrc), this enables using your user and home directory in the container:

```
export GID=$(id -g)
export USER=$(id -u -n)
export HOST_GROUP=$(id -g -n)
export HOST_UID=$(id -u)
```

Unzip installer package:

```
unzip intland-azure.zip
```

There are 2 types of config file examples in the package. One sets the cluster parameters, while the other is responsible for the deployments parameters.

Start container, the home folder is mounted in the container's user's home folder:

```
docker-compose pull
docker-compose run intland-azure
```

Set up deployment environment:

```
export INTLAND_AZURE_AUTH_LOCATION=$HOME/.azure/INTLAND_azure_credentials.json

export SSL_KEY_PATH="$HOME/<path>"
export SSL_CERT_PATH="$HOME/<path>"

export INTLAND_CLUSTER_CONFIG="$HOME/<path>"
export INTLAND_DEPLOYMENT_CONFIG="$HOME/<path>"
```

Create the service principal if it's not yet there:

```
az ad sp create-for-rbac --sdk-auth --name IntlandSP > $INTLAND_AZURE_AUTH_LOCATION
```

Set all parameters in INTLAND_CLUSTER_CONFIG and INTLAND_DEPLOYMENT_CONFIG **before proceeding to the next step.** In case of questions please reach out directly to geza.velkey@intland.com.

Deploy the network, cluster and database (it is heavily recommended to run in **tmux** or screen so if the connection drops out, the script can still complete on the remote machine), deployment takes around 15 minutes:

```
python3 -m intland.azure.cluster deploy_cluster
```

Deploy instance which runs codeBeamer or Intland Retina (3 minutes):

```
python3 -m intland.azure.cluster deploy_instance
```

The DNS name of the instance is written out at the end of deployment logs.

If you want to use the Kubernetes dashboard, it can be turned on (disabled by default):

```
# enable:
python3 -m intland.azure.cluster dashboard_enable
# to get token:
python3 -m intland.azure.cluster dashboard_token
# to disable:
python3 -m intland.azure.cluster dashboard_disable
```

All machines in the VNET can reach the deployment on the requested domain name after the configuration is completed.

Custom configurations

The above script is a best-practice implementation for the most common corporate use-case. There are several configurations in which Intland can help in a consulting manner to create simpler and safer deployments.

Connect partners to the instance

One very common use-case scenario is to connect your external partners to the running codeBeamer/Retina instance. The above created deployment is only available on your company's VNET for security reasons, but there are many ways to connect your peers. As these cases are usually very specific we cannot supply similar out-of-the box solutions as above.

VPN solution

You can connect your partners to the instance (and only to the instance) by creating a new VNET which has peering enabled in the direction of the Intland VNET. Then create a VNET VPN Gateway to which your external users can connect with Point to Site or Site to Site connection. Then create a simple NSG rule in the Ingress subnet (where the https endpoint is located) which allows traffic specifically from this Gateway's subnetwork. Then give access to your partners to this gateway and set up the connection. Your partner sets up a private DNS record with the same name as you to ensure HTTPS connection to the endpoint.
This way all traffic to your instance is sent through via an encrypted tunnel and no public IPs are exposed.

Private Endpoint solution

As the service is located behind an Azure Standard Load Balancer, you can simply create a private link to it. This solution is the best when your partners also use Azure as it directly integrates in the Azure environment and no VPN is required. Your partner can add a private endpoint connection to this Private Link in their own subscription and virtual network and assign the same private DNS name to it.
All traffic goes through Microsoft-only network, so it does not go to the open internet in any way.

Public IP with custom certificates solution

There is another way to access the service by creating a public IP and NSG rules for it and creating a custom certificate which needs all clients to have a private key. This solution is not recommended.

Public IP

Create a public IP and NSG rule so it can access only port 443 in the ingress subnetwork. This allows access from the open internet but only via https. This is not recommended in this setup. This case is covered by Intland SaaS and a deployment can be created by simply filling a web form in a few minutes. This is only recommended if the backend needs to reach your internal resources, but you want to grant access from the open network to your users and partners.