**AUD-20250124-WA0001 (online-audio-converter.com)**

**Transcribed by** [**TurboScribe.ai**](https://turboscribe.ai/?ref=docx_export_upsell)**.** [**Go Unlimited**](https://turboscribe.ai/subscribed?ref=docx_export_upsell) **to remove this message.**

Hello, I'm on very good evening. So let me introduce myself. So myself says I am working with XYZ company from last five years, having total experience, 17 years out of which in DevOps from last six years, I am working as a DevOps engineer.

So let me firstly explain you the use case. I am currently engaged with the US based customer. So the use case for the customer is they have the turbine engines in place.

And if the temperature of the turbine engines goes beyond a certain threshold, there is huge breakdown in the system and huge loss for the customer. So for that, there is a basically application team who have ingested the monitoring solution. So for those applications are running on monolithic and microservices both for monolithic and microservices frontend is react backend few running on Python, few on Java, few on .NET and database on cloud and few on on-premises.

So this is what all architecture for the application. So suppose like any vendors want this solution. So there is a separate application team for the 10 peoples.

And for those teams, we are supporting from the DevOps end. So I am acting as an architect as well as implement a role here. So when we talk about my roles and responsibility, so once we develop the high level diagram and get it approved by the Microsoft architect and customer architect, once that is approved, then me along with my team is to implement those high level design solutions, create the LLD low level design, so basically for DevOps best practices and solutioning implementation, we have just chosen ISA tool Terraform.

So when we talk about Terraform, so we have written Terraform generic modules considering dynamic block for each map of object and then optional attributes. So all the optional attributes of level on the registry, we have integrated those all with the dynamic block and for each iteration. So if any user want to create some optional attribute in next near future, they not need to go and change in the module code.

It's already present with all the generalization of the code. So that generic, we have created the modules starting from management group to subscription, its association. Coming to RG, VNet, Subnet, definitely peering VPN gateway.

And it comes to load balancing and compute workloads, Windows, Linux. And on top of that load balancing, internal load balancer, external load balancer with the public IP application gateway for multi-site deployment front door for the regional services and traffic manager. These services too, we have written a generic modules.

Apart from this, you have when we talk about SZ, NSZ, firewalls, UDR, routes. So for each and every component, we have generic modules written apart from monitoring solution, backup solution, DR solutions, everything generic modules and that is stored in a multi-repository concept for each modules, resource block modules, we have a separate repository with a versioning system placed on the Azure repository. And suppose now from here, we need to provide the solution, at least the three environment for the customers.

One is sandbox environment on just testing environment and then it comes to development or you can pre-prod or prod environment. So in sandbox, suppose we just need to give a testing subscription. So we are calling the management group module and then calling subscription module, associating it together, putting right dependencies through a pipeline.

We are deploying this setup with a state file on remote blob storage with customer managed key enabled in CMK. And suppose for the development, we are calling all the modules based on the requirements of the customer management group subscription, its association to RG, VNet, subnet pairing, whatever accordingly we are calling just creating a parent module or putting right dependencies, creating a generic TFRs and putting this code in a trunk-based branching strategy in Azure repository. So basically once the pipeline is set in the multi-stage YML pipeline, we are using here Azure DevOps pipeline.

So once we have the pipelines setup is done, so what generally the DevOps process is, suppose one new VM needs to be added, then firstly we will take the clone of existing repository for which we have created the parent job, creating a feature branch, added a new VM in the TFRs, using that in a feature branch and then raising a PR as soon as we raise a PR automatic a pipeline will run and in that pipeline firstly it will take the sanity check stage in that linting process, it will check TF lint tool is there that will check the linting process of the code, there is a check ops tool for vulnerabilities and there is a truffle hog for security scanning, TF sec for vulnerability scanning. So these all are the like security tool and linting process tool. Apart from this, we have integrated chef inspect tool for the test cases integration.

So if these all test passes and goes to the plan, then only approvers can able to merge into the main branch. And once that is merged to the main branch, automatic pipeline will run it look for all the sanity check, which we have discussed for the check ops, TF sec, truffle hog and go for the plan stage and it went for the manual validation stage, which is a gen test job and then it go and once that is approved, it get deployed on the development environment and new VM get added to the inventory of the development environment and same with how it goes for the productions to for the infrastructure automation. So this is a high level, either it's a monolithic application or a microservices application.

This is where we are handling end to end infrastructure provisioning and when it comes to the microservices application. So there too, like we have AKS, ACR, Key Vault, so Bassian, yeah, definitely Bassian. So these all, we have a generic models created, put it in the multi repository with a versioning, calling this and creating the landing zone for the, when it is clustered at one go with this DevSecOps pipeline with the proper branching strategy, that is trunk based branching strategy.

And once we have the infrastructure in place on top of that, we need to deploy the application. So when we talk about the monolithic application, so we need to install the middleware, so for the middleware installation, either it's a Windows IS based server or Linux, it's a Nginx, might be on an operating system, it might be Java application, it's a Tomcat Apache, but depending upon React application, it's a Nginx, .NET application might be an IS based server. So depending upon the nature of application, we are installing that middleware on top of the machines.

So once that is installed and that we have written in the Terraform module itself, if the machine type is Windows or Linux based on that, it will install respective with the custom data script in the Terraform modules, it will install the required web servers. And once that is installed on top of that, we need to deploy the application. So for that to be a proper branching strategy, that is GitFlow branching strategy, so we have, definitely we have a hot release branch, there's a hotfix branch, there's a feature branch, dev branch, and then main branch, or you can say production branch, and there's a hotfix and release branch too.

So for this too, we have a separate pipeline. So we have a, suppose a new feature needs to be added. So we have a, the first developer will clone it, create a feature branch.

And then after making a change, push it to the feature, raise the PR, pipeline will run, it will generate the, check the quality of the codes with the Sonar queue. And might be sync tool or check marks tool for scanning the vulnerability, third part is for Java, it's a form.xml, so all dependencies. Generally be used to not call from Google, generally be used to create the artifactory like JFrog, Nexus or Azure artifact as a feed, and used to put all dependencies there in repository and calling at runtime to install the third-party dependencies, and once this is build and build is pass, then only the approver can able to merge that to the development branch, and once that is done, then again, automated pipeline will run and that check the quality of the code with the Sonar queue tool integrated.

That checks the quality of the code with the custom rules and few built-in rules. And there is a rule integrated, a policy integrated in the pipeline itself. If that, if the quality of the code is more than 80%, then only it will pass, else it will fail, and then we have integrated test cases might be for Java.

We have used the J unit and one of the test cases and everything is passed. For vulnerability scanning, we have check marks tool integrated. If any higher and medium vulnerability, it will break the pipeline and send notification to the development team to fix that vulnerability scanning.

And once the artifact is generated, it is published to the Azure artifacts. And from there we have automated deployment test region, dev region and test region. And that is once done, it is merged to the production or main branch and then automated deployment is still test.

And then from there, it will go to the QA branch where approvers in place. Once that approved, it will deploy to QA and it goes to the production where businesses, stakeholders, and unit heads, DevOps, development, quality assurance and client stakeholders. Once they approve, it will get deployed to the respective production environment and notification will roll back.

And there is a separate pipeline for the rollback with the artifact versioning. If you want to, you know, rollback it. So in the DevOps best practices, if something is, you know, deployed wrong, the first thing we need to do is rollback it to save the business losses and all.

So what we will do, we will just trigger the rollback pipeline with the right artifact versioning and once it deployed and it will make the site up. So this is all rollback. And similarly, we have the backup strategy, GR.

Yeah, definitely for monitoring, we have a setup for Prometheus and Grafana and Datadog for the application monitoring and matrices. Our backup, definitely on Azure, we are relying on the backup for the infrastructure and for the application. Definitely we are taking back of the code on the GitHub and then artifact itself is a backup.

We can redeploy the previous version of artifact and make the system up when it is needed. So, yeah, definitely we have for the governance for Azure policies, built in and customs we have integrated CIS benchmark policies. We have integrated to follow the GDPR rules and regulations.

So those are compliances and reports getting generated by Bigley and sending across the DL for a support team to get a fix of and rid of any issues or noncompliance attributes of level on the dashboards for the Azure policies. Apart from this, when it comes for the Azure, our customer is getting migrating the stuff from monolithic to microservices due to cost, time and quality and downtime feature. So they are just, you know, moving few non-critical system as of now to Kubernetes.

So again, the same we have integrated, we have ISQL Terraform through that. We have written all those models and. Creating the landing zone with the Azure DevOps pipeline and multistage deployment with proper DevOps, DevSecOps practices.

And once the landing zone is deployed for the Kubernetes and then we need to write the manifest file, once we have written the Docker file, we have generated the image and push it to the ACR and there before pushing, we are scanning with the TriviTool and Sonarchy in place for code quality and all things are running on containers. And if everything pass, push it to the ACR with the image new versioning. And from there, we are writing the manifest files that is deployment.yml for, you know, pulling that image and running in deployment and replicas request limits and all the features we can include in that deployment.yml. And there for internal load balancing, we have the service type cluster IP.

And then on top of that, we are using the services ingress and entry point is the application gateway. So we have enabled the AGIC ingress application gateway ingress controller that give the ingress and we can enable the ingress with the application gateway and we can enable the BAF and all the firewall policies. Entry point is that from application gateway come to ingress through controller and where we have written the ingress rule, it will direct the traffic to the internal cluster IP services based on the rule and from there, the load will transfer from the pod, whatever the pod we are running in the deployment and might be have the config.yml, secret.yml, pb.yml, pvc.yml. So lots of YML we have written and in the near future, we are just going to plan and use a packaging tool, a simple POC, we did a Helm chart.

So what we did is we are generating and creating the values, templating at a runtime when passing the values.yml and it generates the template and all the manifest files at runtime. So we're in a phase in next PI process improvement plan. We're going to use Helm for this integration.

And as of now, we are using Azure DevOps pipeline in multistage deployment with the DevOps best practices, but in near future, we are just going to GitOps. So for that too, we are just trying to integrate Argo CD. So this is still in R&D phase.

We are doing R&D on Argo CD and we will for the state management and clarity on how deployment is getting created. So minute management and monitoring, we will be going to implement Argo CD in next quarter and for monitoring, definitely we have Prometheus and Grafana, Prometheus collecting the matrices and sending the database of Prometheus and then Grafana few dashboards we have created for cost optimization, monitoring and then any security vulnerabilities kind of role. So this is all over what exactly we are working on.

Definitely DevOps is never ending process. We are looking something every day new to automate the soliciting, reporting, onboardings. So new initiatives every year now in every sprint we are planning and trying to automate end to end setup, either it's application, either infra, monitoring, observability.

So everything we are working in that area. So every PI planning is to target us improvement in our existing system, a new soliciting, design and implement. This is how we are working on engineering side and making the DevOps process more smooth and reliable.

So yeah. Thank you.

**Transcribed by** [**TurboScribe.ai**](https://turboscribe.ai/?ref=docx_export_upsell)**.** [**Go Unlimited**](https://turboscribe.ai/subscribed?ref=docx_export_upsell) **to remove this message.**