**Wiz Support Technical Exercise Intro / Overview**

Welcome to the Wiz Support Technical Exercise! This is your opportunity to demonstrate compelling technical proficiency and your domain expertise in modern cloud architecture.

You can choose any of the “Big 3” CSPs you are most comfortable/familiar with (either AWS, GCP, or Azure).

**The Presentation**

You will be presenting your solution (leveraging the provided template) to 2-3 Wiz panelists via Zoom. The panel will assess what you built, your methodology, challenges faced & the quality of your overall presentation. As this a Support Engineering role, please make sure to document how you went about troubleshooting any issues you encountered, including your debugging logic and tools/commands used.

Excellent presentations will:

• Incorporate a mix of slides and live walkthrough

• Evidence the proper function of the infrastructure

• Discuss your approach to the build-out, including challenges and adaptations

• Detail your grasp of weak configurations and their potential consequences **(Bonus)**

**Exercise Layout / Timing**

When you are ready, we will schedule you with a Wiz expert panel. Plan for your initial presentation to take about 15 minutes. In the following 30 minutes, our panel will be asking questions about what you built, as they will be leveraging a scoring rubric and need all the details. With the remaining 15 minutes, we can have a discussion about how Wiz works and answer any questions you may have about our solution.

**A screenshot of a computer

Description automatically generatedTechnical Exercise Requirements/Details**

Environment Requirements:

* **Web Application Tier:** A containerized web application deployed on Kubernetes.
* **Database Tier:** A database server set up on a VM in a public subnet.
* **Storage Tier:** Cloud object storage configured as anonymous public readable.

**Kubernetes Application Exercise**

Your web application will be publicly accessible and set up to communicate with the database server. The database backups should be automated and stored in the public-readable cloud object storage.

At the end of basic setup, you should have a **working web application** which can be accessed from the web, and a VM instance running MongoDB. Before the presentation begins, ensure that every component of the exercise is operational and ready for demonstration.

* **Database:** Set up a VM using an outdated Linux version. On top of this VM, install an outdated database server. (DB example – MongoDB). Configure the VM to allow SSH connections from the internet. Configure the Database and/or associated networking to only allow connections to the Database from the **applications deployed to your Kubernetes cluster.**
* **Database authentication**: Ensure the database is configured for local authentication so you can construct a database connection string.
* **Highly Privileged DB VM**: Configure the VM in a way that it is granted overly permissive CSP permissions.
* **Object Storage**: Create a cloud object storage resource which will store the database backups. Modify the permissions on the cloud object storage resource to allow public read access to the backups stored within, as well as listing contents.  
  *You will be asked to validate during the review that the backup is accessible via an external URL.*
* **DB Backups**: Create automation which regularly backs up the database(s) to the created bucket.
* **Kubernetes Cluster**: Deploy a Kubernetes cluster to host a containerized web application.
* **Containerized Web Application**:
  + Build and deploy a containerized web application to the Kubernetes cluster. You can develop your own, utilize open-source solutions, or try this sample: <https://github.com/jeffthorne/tasky>
  + Ensure the container employs database authentication. This typically uses a connection string format.
  + Confirm the built container image includes a file named "wizexercise.txt" with content.
* **Public Access**: Set up the containerized web application to be reachable from the public internet.
* **Container Admin Configuration**: Configure the web application container to run with cluster-admin privileges.

**(Extra Credit) DevOps Exercise**

While optional, you also have the opportunity to further showcase your skills by leveraging modern DevOps practices to automate the building of your cloud infrastructure and the deployment of your application.

* **VCS/SCM**: Push your code to a VCS/SCM of your choice (GitHub, Gitlab, Azure DevOps, etc.)
* **CI Pipelines**: Setup two CI pipelines:
  + One CI pipeline to securely deploy your cloud infrastructure using IaC (Terraform, CloudFormation, etc.)
  + One CI pipeline to build & push your containerized application to a container registry of your choosing.

**Additional Resources:**

* CI Systems
  + Github Actions - [Docs](https://docs.github.com/en/actions)
  + Jenkins - [Docs](https://www.jenkins.io/doc/book/pipeline/)
  + CircleCI Pipeline - [Docs](https://circleci.com/docs/)
* Kubernetes RBAC Documentation: [Kubernetes RBAC Documentation](file:///Users/erik.papir/Downloads/SE%20Techical%20Exercise%20Template/●%09https:/kubernetes.io/docs/reference/access-authn-authz/rbac)
* Infrastructure-as-Code tooling for Kubernetes deployment:
  + Terraform Kubernetes Provider: [Terraform Kubernetes Provider Documentation](https://registry.terraform.io/providers/hashicorp/kubernetes/latest/docs)
  + Helm – The Kubernetes Package Manager: [Helm Official Site](https://helm.sh/)