European Union Agency
For Cybersecurity

Ethnikis Antistaseos 72
Chalandri 15231 | Attiki | Greece
Tel: +30 28 14 40 9711
E-mail: info@enisa.europa.eu
www.enisa.europa.eu

EU-CSI DISASTER RECOVERY POLICY

TABLE OF CONTENTS

EXECUTIVE SUMMARY

This document outlines the measures that have been taken and proposes preventive actions to **ensure the availability, resilience, and recoverability of the EU Cybersecurity Index (EU-CSI) platform**. The scope of this document is to present a comprehensive disaster recovery plan that will minimize downtime and ensure the integrity and availability of critical data, while maintaining business continuity and meeting compliance and regulatory requirements.

This document will be evaluated by relevant stakeholders, and some of the proposed action plans may be validated or enhanced based on their alignment with the organization's policies and objectives. In this Executive Summary, we present a high-level summary of the contents of this document, while in the remainder of the document, these concepts will be explained in more technical detail. The main aspects regarding our disaster recovery approach are summarised below.

Infrastructure: We outline the resources that are used to host the application. We identify the critical resources and components which serve the application and state the preventive measures that have been taken in a disaster scenario as well as the recommended extra-step solutions that should be taken, leveraging cloud-native solutions, to ensure further resilience. A set of design and configuration decisions have been made to support this process, including the Kubernetes cluster configuration and management and the setup of the cluster for high availability. Other critical parts of the infrastructure deal with data storage and retrieval (database modules), as well as email communication within the context of the EU-CSI platform. For those parts, we have set a plan for scheduled (automated) and manual (on-demand) backup of data, as well as redundancy based on the availability zones model of the cloud provider. More technical details can be found in *Section 1: Infrastructure*.

Application: For the application architecture, we follow the modular, containerised microservice paradigm. We ensure high-availability and resilience of the updated containers by utilizing a cloud-based container registry. New features and updates on the application functionalities are shipped following rigorous and tested Continuous Delivery (CD) pipelines that deploy each release in the staging and production environments, thus supporting the application's stability during a disaster recovery scenario. Application data are stateless and are therefore stored, retrieved and backed up as described in the *Infrastructure* section. More technical details on the Application aspects can be found in *Section 2: Application*.

Monitoring and alert: We detail the setup selected for continuous, real-time and proactive monitoring of the operation of the application. We define a number of metrics to be measured and methods and tools, such as Dashboards and email notifications, to monitor and take actions on these metrics. We use specialized dashboards for the Kubernetes cluster and Database health, as well as alerting mechanisms for cluster availability or failures and application logs. More technical details can be found in *Section 3: Monitoring and alert*.

Change management: Of critical importance is the definition and application of change management processes to ensure stability and reliability of the platform. We address change management in various levels: a) application changes, with the use of code versioning and quality assurance tools, b) infrastructure changes, with the automations based on the Infrastructure as Code paradigm, and actions plans to be based on the Resource Inventory and Resource Manager. More technical details can be found in Section 4: Change management.

Disaster recovery drills: We define regular disaster recovery drills to ensure the resilience and reliability of the application. By frequently testing and validating recovery procedures, any discrepancies or issues within the established plan can be identified and addressed promptly, minimizing the risk of prolonged downtime in the event of an actual disaster. Moreover, these drills assist in pinpointing bottlenecks and weak points within the application and infrastructure, thus enabling optimization of the recovery process and further enhancing the system's overall performance. We detail concrete scenarios with a series of well-defined steps to be executed, for various critical cases, including application data with database backup and restore, infrastructure availability with resource management templates, and container registry image restoration and security. More technical details can be found in *Section 5: Disaster recovery drills.*

1. INFRASTRUCTURE

1.1 EUCSI ARCHITECTURE

The components (and their corresponding technologies) of the EU CSI are depicted in the following figure.

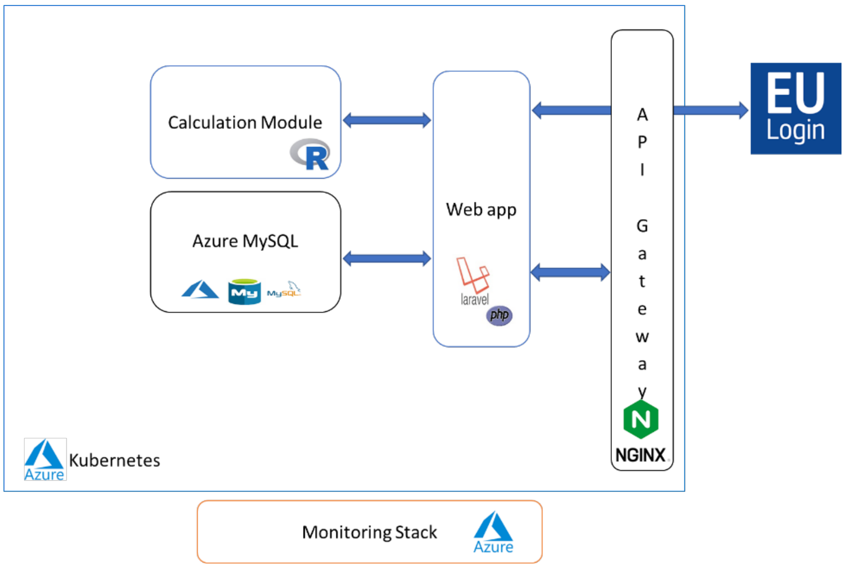


Figure 1 High-level architecture of EU CSI

Component Name	Description	Technology
Web UI	The graphical User Interface provided to the end-user in the form of a web application.	Laravel Blade[1]
Core	The EU CSI Core is a modular component comprising: • A Rest API • The Index Calculation Module • The Data Ingestion Module • Notification Module	PHP Laravel[2]
Index Calculation Module	The CoinR module is used for calculating the index and for producing reports. The module is exposed internally as a Rest API, used by the EU CSI Core.	COINR[3]
Gateway	Assuming the presence of multiple nodes (that could be also created on demand) on the Kubernetes cluster to accommodate increasing service demand levels, this module will also be responsible for the system's dynamic load balancing in response to incoming requests by distributing the load evenly to available nodes to achieve (enabled by Kubernetes) high system availability and performance with optimized use of resources.	Nginx[4]
RDBMS	The data store is based on a Relational Database System. All information that needs to be persistent will be stored in this DB.	Azure Mysql[5]

[1] https://laravel.com/docs/5.1/blade
[2] https://laravel.com/
[3] https://knowledge4policy.ec.europa.eu/composite-indicators/coinr_en
[4] https://www.nginx.com/
[5] https://azure.microsoft.com/en-us/services/mysql/#overview

	Relying in an Azure fully managed MySQL database provides scalability and high availability of the data.	
Identity & Access Management	All users login through the ECAS EU login system. However, in order to manage the access control (authorization) an Identity and Access Management module is required (implemented in the EU CSI Core).	EU Login PHP client
Monitoring Stack	The monitoring stack collects security, audit and application logs from the various components and provide a UI with search and filter capabilities.	Azure

1.2 KUBERNETES CLUSTER

The application utilizes a Kubernetes managed cluster in Azure AKS for several compelling reasons that significantly benefit the development, deployment, and management processes. These reasons include:

- **Simplified management**: Azure AKS handles the management and maintenance of the Kubernetes control plane, allowing the team to focus on developing and deploying applications instead of dealing with the complexities of Kubernetes infrastructure. This reduces the operational overhead and enables the feature to deliver faster.
- **Scalability**: AKS provides easy scaling options, both for the cluster and for individual applications. The cluster autoscaler and the horizontal pod autoscaler enable us to quickly adjust resources based on demand, ensuring optimal performance and cost efficiency.
- **High availability**: By leveraging Azure Availability Zones and multiple node pools, we can ensure high availability and resiliency for our application. This helps minimize downtime and provides a better user experience for our customers.
- **Integration with Azure services**: AKS is seamlessly integrated with various Azure services, such as Azure Active Directory, Azure Monitor, and Azure Container Registry. This simplifies authentication, monitoring, and container image management, streamlining our development and operations workflows.
- **Security and compliance**: Azure AKS provide built-in security features, such as role-based access control (RBAC), network policies, and Azure Private Link support. These features, along with Azure's compliance certifications, help us maintain a secure and compliant application environment.

1.2.1 Current AKS Cluster High Availability Setup

- **Stateless Application Architecture**: All application data is stored in a managed Azure database, enabling the implementation of a stateless application architecture. This design choice allows for increased scalability and resiliency.
- **Autoscaling**: It has been configured for the AKS cluster, allowing for the automatic adjustment of resources based on demand. This feature helps maintain optimal performance and high availability during varying workloads.
- **Multi-Zone Availability Configuration**: A two-node resiliency cluster has been deployed, spanning across different Azure Availability Zones. This configuration ensures that the application remains available even if an entire datacentre or zone experiences an outage.
- **Deployment and Rolling Update Strategy**: To maintain high availability, the application components have been encapsulated in Kubernetes Deployments. The optimal rolling update strategy has been selected, enabling seamless and uninterrupted updates to the application without affecting its availability.

The current infrastructure specifications are the following:

- Azure Kubernetes Service (AKS) with the following specs:
  - Multiple availability Zones
  - Node size: Standard_D2_v2 (7GiB Memory, 2cores)
  - Autoscaling enabled (if autoscaling is not enabled, at least two nodes are required)
- Azure Database for MySQL
  - Resource type: Flexible server
  - MySQL Version: 8
  - Workload type: For small or medium size databases

- Compute + Storage: General Purpose, D2ds_v4
- Enable High availability: checked
- DB Backup enabled with 30 days retention period

The Cluster comprises of the following services/containers:

Service	Container name	Image (current)
EU CSI Core	csi-container	eucsistg.azurecr.io/csi_app:v1.1.6
Gateway	nginx-container	eucsistg.azurecr.io/csi_nginx:v1.1.6
Calculation Module	calcmodule-container	eucsistg.azurecr.io/calcmodule:v1.0.0

1.2.2 Proposed Enhancements for Disaster Recovery

1.2.2.1 Utilizing AKS Backup Extension:

In terms of **functionality**, the Azure Kubernetes Service (AKS) backup mechanism provides a powerful and flexible solution for backing up and restoring Kubernetes workloads and data in AKS clusters.

One of its key features is the ability to configure scheduled backups for cluster state and application data, which includes persistent volumes based on the CSI driver-based Azure Disks. This ensures that data remains secure and up to date. Additionally, AKS backup allows for granular control, enabling to choose a specific namespace or an entire cluster to back up or restore. This caters to various use cases such as operational recovery, cloning developer/test environments, or cluster upgrade scenarios. Furthermore, the backup solution supports backing up Kubernetes workloads and data stored in Persistent Volumes, storing workloads in a blob container, and creating Disk Snapshots for Disk-based Persistent Volumes.

When it comes to **cost**, the AKS backup mechanism is designed to be cost-effective and efficient. Charges are incurred only for the retention of backup data stored in the blob container and for the Disk-based persistent volume snapshots stored in the resource group within your Azure subscription. By utilizing incremental snapshots, AKS backup minimizes costs as these snapshots are charged per GiB of storage occupied by the delta changes since the last snapshot. This approach reduces overall storage consumption and associated costs, as you only pay for the actual changes in your data.

1.3 DATABASE

In the following sections, DR preventive measures, which are **currently implemented,** are described. No further action is required.

1.3.1 Azure Managed Backup

1.3.1.1 Automated Backups

Daily backups of database files are automatically performed by the service, along with continuous transaction log backups. Backups are retained for a period of 35 days. The database server can be restored to any point in time within the backup retention period. The time required for recovery is dependent on the size of the data to be restored and the time needed to perform log recovery. the service backups can be configured as geo-redundant. The replication of server backup data files is facilitated in the primary region's paired region, providing regional resiliency. Geo-redundant backup storage ensures a durability of at least 99.99999999999999% (16 nines) for objects over a given year.

1.3.1.2 Manual Backups

In addition to the automated daily backups provided by Azure Managed Database, it is essential to incorporate manual backups, which will be stored in blob storage for longer durations. These manual backups complement the native backup capabilities and serve a critical role in testing and validating our disaster recovery (DR) policy.

Manual backups stored in blob storage allow us to perform regular DR drills and ensure that our recovery procedures are effective and well-documented. These drills help identify potential gaps or issues in the recovery process, enabling us to refine our DR policy and improve its overall efficiency. Furthermore, having manual backups in blob storage extends the retention period beyond the 35 days offered by Azure Managed Database, providing an additional layer of data protection in case of long-term data corruption or loss.

The process is well documented by Azure community and details can be found here.

1.3.1.3 Security aspects of Azure blob Storage

Storing manual backups in Azure Blob Storage provides several advantages in terms of security, as opposed to keeping them within the infrastructure. Azure Blob Storage is designed with security in mind, offering robust mechanisms to protect and control access to stored data.

One key security feature of Azure Blob Storage is data encryption at rest. All data stored in Azure Blob Storage is automatically encrypted using Azure Storage Service Encryption (SSE), which ensures that data remains protected without any additional effort on your part. Furthermore, Azure Blob Storage allows for the implementation of customer-managed keys for added control over encryption keys.

Access control is another crucial aspect of security. Azure Blob Storage supports Azure Active Directory (AD) based authentication and role-based access control (RBAC), enabling granular permissions management. This ensures that only authorized personnel can access the stored backups, reducing the risk of unauthorized access or data breaches.

Azure Blob Storage also offers advanced threat protection, monitoring, and auditing features that help detect and mitigate potential security risks. With Azure Security Center integration, you can monitor and assess the security posture of your storage accounts, providing insights into potential vulnerabilities and offering recommendations for improvement.

In addition, Azure Blob Storage is designed to be resilient against infrastructure failures. It ensures data durability and high availability by automatically replicating stored data across multiple facilities within a region or even across multiple regions, depending on the selected redundancy option. This replication strategy protects against data loss caused by hardware failures, natural disasters, or other unexpected events.

1.3.2 Zone Redundant Replica for High Availability

The database service is deployed in high availability mode, which deploys primary and standby servers in two different availability zones within a region. Zone redundant high availability protects from zone-level failures and helps with reducing application downtime during planned and unplanned downtime events. Data from the primary server is synchronously replicated to the standby replica. During any downtime event, the database server is automatically failed over to the standby replica.

1.4 EMAIL SERVER

The current VM which serves the functionality of the webserver does not contain any application data, but only configuration. However, the following measures can provide additional peace of mind and protection, ensuring the continuity of the email services in the event of an issue or disaster.

1.4.1 Azure VM Backup

Recommendation: It is proposed to utilize the native-Azure tools for a full system backup with a retention policy of 7 days.

2. APPLICATION

2.1 CONTAINERIZATION

The application, being fully containerized, benefits from the inherent advantages of containerization, such as the creation of consistent runtime environments, the ability to predict and handle future increased loads, and efficient versioning and rollback capabilities. The Azure Kubernetes Service (AKS) cluster is responsible for orchestrating the deployment of our application workloads, which are sourced from the Azure Container Registry (ACR). Consequently, establishing a well-defined disaster recovery process is vital to maintain the uninterrupted operation and availability of the application, safeguarding it against unexpected events or failures.

2.2 CONTAINER REGISTRY

2.2.1 High Availability and Resiliency of ACR

Azure Container Registry (ACR) is built on top of Azure Storage, which is designed to be highly available and durable. When you store your container images in ACR, they benefit from the built-in resiliency features of Azure Storage. By default, Azure Storage uses locally redundant storage (LRS) to store multiple copies of your data within the same datacenter. For each image stored in ACR, Azure creates three replicas within the same datacenter, ensuring that data remains available even if there are hardware failures or other issues affecting one or more replicas.

2.3 CONTINUOUS DEPLOYMENT (CD) PRACTISES

2.3.1 Build and Push Pipelines

The process of building the application images is fully automated using Jenkins. In case of an accidental image mismanagement in the container registry surface, the pipelines provide quick resolution and minimal downtime. Furthermore, pipelines are configured to include unit tests and Xray scanning of the images to minimize the risk of faulty or security risky images. These measures ensure that only high-quality and secure images are deployed in the production/staging environment.

2.4 APPLICATION DATA

During the design process, special care has been taken to ensure the stateless nature of the application. As a result, all the application data is stored inside the database and its backup strategy in case of a disaster recovery scenario is described in previous sections. The AKS cluster's persistent volumes store temporary reports generated by the application. Although these reports can be regenerated if required, the AKS backup mechanism is proactively enabled to ensure rapid recovery and minimal data loss in the event of a failure.

3. MONITORING AND ALERT

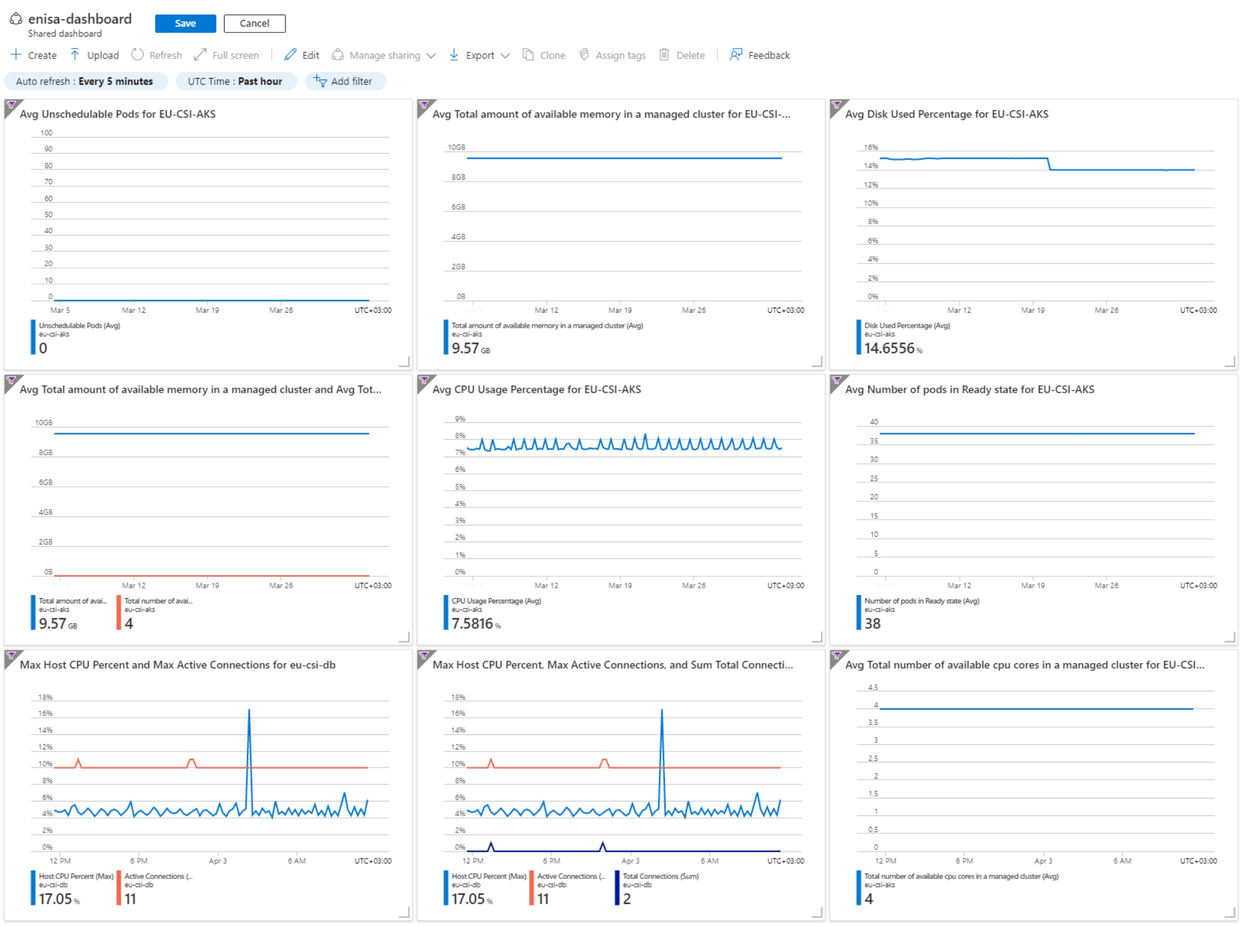
3.1 MONITORING DASHBOARDS

Dashboards provide a centralized location for monitoring health and performance of the various infrastructure components and more specifically, the EKS cluster and the Azure managed Database. By monitoring key metrics and logs, potential issues can be addressed before they become critical and as a result, the risk of a major disaster is reduced significantly. Also, new opportunities for optimization and cost reduction can be identified. By reducing unnecessary resource usage, the risk of a disaster caused by overutilization can be minimized.

3.1.1 Kubernetes Cluster Performance

The following list contains the key metrics that are monitored regarding the health and operation of the Kubernetes cluster:

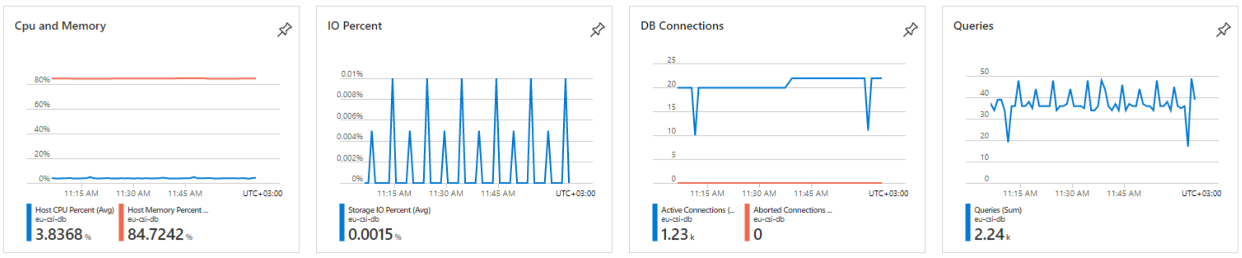
- Average number of Schedulable Pods
- Average total amount of available memory in a managed cluster
- Average Disk Used percentage.
- Average CPU usage percentage
- Average number of pods in ready state
- Max Active Connections
- Total number of available CPU cores.


enisa-dashboard
Shared dashboard
Save
Cancel
Create
Upload
Refresh
Full screen
Edit
Manage sharing
Export
Clone
Assign tags
Delete
Feedback
Auto refresh : Every 5 minutes
UTC Time : Past hour
Add filter
Avg Unschedulable Pods for EU-CSI-AKS
Unschedulable Pods (Avg) eu-csi-aks 0
Avg Total amount of available memory in a managed cluster for EU-CSI-...
Total amount of available memory in a managed cluster (Avg) eu-csi-aks 9.57 GB
Avg Disk Used Percentage for EU-CSI-AKS
Disk Used Percentage (Avg) eu-csi-aks 14.6556 %
Avg Total amount of available memory in a managed cluster and Avg Tot...
Total amount of avai... eu-csi-aks 9.57 GB
Total number of avai... eu-csi-aks 4
Avg CPU Usage Percentage for EU-CSI-AKS
CPU Usage Percentage (Avg) eu-csi-aks 7.5816 %
Avg Number of pods in Ready state for EU-CSI-AKS
Number of pods in Ready state (Avg) eu-csi-aks 38
Max Host CPU Percent and Max Active Connections for eu-csi-db
Host CPU Percent (Max) eu-csi-db 17.05 %
Active Connections (... eu-csi-db 11
Max Host CPU Percent, Max Active Connections, and Sum Total Connecti...
Host CPU Percent (Max) eu-csi-db 17.05 %
Active Connections (... eu-csi-db 11
Total Connections (Sum) eu-csi-db 2
Avg Total number of available cpu cores in a managed cluster for EU-CSI...
Total number of available cpu cores in a managed cluster (Avg) eu-csi-aks 4
UTC+03:00

3.1.2 Database Health Dashboard

The following list contains the key metrics that are monitored regarding the health and operation of the managed database:

- CPU and Memory
- IO Percent
- DB Connections
- Queries


Cpu and Memory
Host CPU Percent (Avg) eu-csi-db 3.8368 %
Host Memory Percent ... eu-csi-db 84.7242 %
IO Percent
Storage IO Percent (Avg) eu-csi-db 0.0015 %
DB Connections
Active Connections (... eu-csi-db 1.23 k
Aborted Connections ... eu-csi-db 0
Queries
Queries (Sum) eu-csi-db 2.24 k
UTC+03:00

3.2 ALERTING MECHANISMS

3.2.1 Cluster Alerts

Alerting mechanisms have been established for specific cluster metrics, based on predefined thresholds. When any of these monitored metrics surpass the designated threshold, an automated email notification will be sent to the relevant stakeholders. This approach ensures timely awareness of potential issues and facilitates prompt action for maintaining the health and stability of the Kubernetes cluster.

To further elaborate, the monitored metrics for the **Kubernetes cluster** and their corresponding threshold values are as follows:

- node_disk_usage_percentage > 40
- node_cpu_usage_percentage > 80
- CpuExceededPercentage > 60 (per pod)
- node_memory_working_set_percentage > 80
- nodesCount < 2

The corresponding list for the **database**:

- abort_connections> 2
- cpu_usage>60

The corresponding list for the **email server**:

- cpu_usage>80
- Data Disk IOPS Consumed Percentage > 95.

3.2.2 Application Log Monitoring

A refined log analysis mechanism has been developed and deployed to continuously scrutinize the logs of the Laravel pod. This system is engineered to detect instances of HTTP 500 errors, indicative of service unavailability. Upon identification of such an event, the relevant stakeholders will be duly notified,

4. CHANGE MANAGEMENT

Change management plays a critical role in ensuring the stability, reliability, and security of both the application and the underlying infrastructure. Effective change management involves the implementation of well-defined processes, tools, and communication strategies to manage, track, and coordinate modifications to the application and infrastructure components.

4.1 APPLICATION CHANGE MANAGEMENT

Application change management focuses on managing modifications to the application code, configuration, and dependencies. This includes tracking new features, bug fixes, performance improvements, and security updates.

Bitbucket is utilized as the hosted version control system for the project, effectively managing the application code and configuration files. This **eliminates** the need for manual backups and provides an accessible history of changes, which is essential in disaster recovery scenarios.

Application developers work on their own branches in the repository, ensuring that changes are isolated and easily tracked. **Pull requests** are used to merge these changes into the main line of development, following a review process. This approach maintains code quality and prevents potential issues from affecting the main line of development.

4.2 INFRASTRUCTURE CHANGE MANAGEMENT

In the initial stages of the project, the primary focus was on rapidly deploying the application, which led to the adoption of a manual approach to infrastructure management using Azure portal UI. This approach involved building on top of existing resources, such as subscriptions, instead of importing them. While this choice allowed for faster implementation, it also lacked the benefits of an Infrastructure as Code (IaC) solution. Despite the simplicity of the infrastructure, a disaster recovery plan is essential for any project. The following action plan will be implemented immediately in future iterations and will be composed of two main components, a **Resource Inventory** and utilization of **Azure Resource Manager.**

4.2.1 Resource Inventory

An Azure resource inventory is a comprehensive and structured documentation of all resources within an Azure subscription, which aims to provide a clear understanding of the current Azure infrastructure and its dependencies. This inventory serves as a foundation for tracking changes, updates, and the overall management of the Azure environment. It plays an essential role in planning and executing a disaster recovery strategy. The following elements would be included:

- Azure Resource Groups
- Compute Resources
- Azure Storage Accounts
- Azure Networking Components
- Azure Managed Services.

4.2.2 Azure Resource Manager

Azure's native capability to export and test Resource Manager Templates directly is a viable solution for managing infrastructure changes. This approach leverages the built-in functionality provided by Azure, ensuring seamless integration with the platform, and potentially reducing complexity and overhead associated with third-party tools.

In the context of disaster recovery planning, ARM Templates can provide several benefits:

- **Consistency**: Utilizing ARM Templates ensures that Azure resources are created and configured consistently, reducing the risk of configuration drift and human error. This consistency is essential for minimizing downtime and facilitating a smooth recovery process.
- **Versioning**: ARM Templates can be stored in a version control system, such as Git, which allows for tracking changes, rolling back to previous versions, and maintaining a history of the infrastructure. This versioning capability is crucial for managing changes and understanding the impact of infrastructure modifications on the application.
- **Automation**: ARM Templates enable the automation of deployment and management of Azure resources, reducing manual intervention and increasing operational efficiency. This automation can significantly decrease the time required for recovery from a disaster, ensuring that the application is operational as quickly as possible.
- **Reusability**: ARM Templates are modular and can be reused across multiple environments or projects, promoting infrastructure standardization and adherence to best practices.

5. DISASTER RECOVERY DRILLS

Regular disaster recovery drills play a critical role in ensuring the resilience and reliability of the application in question. By frequently testing and validating recovery procedures, any discrepancies or issues within the established plan can be identified and addressed promptly, minimizing the risk of prolonged downtime in the event of an actual disaster. Moreover, these drills assist in pinpointing bottlenecks and weak points within the application and infrastructure, thus enabling optimization of the recovery process and further enhancing the system's overall performance.

5.1 MANUAL DATABASE BACKUP AND RESTORE

Objective: To validate the effectiveness of the manual backup process and the ability to restore the Azure managed database to a specific point in time.

Steps:

1. Select a specific point in time for which a manual backup is available in Azure Blob Storage or take a manual snapshot.
2. Simulate a disaster scenario by creating a new Azure managed database instance.
3. Retrieve the manual backup from Azure Blob Storage corresponding to the selected point in time.
4. Restore the Azure managed database using the retrieved manual backup.
5. Verify that the restored database contains the expected data and is functioning correctly.
6. Evaluate the time taken to complete the restore process and identify any bottlenecks or areas for improvement.

5.2 ARM TEMPLATE VALIDATION AND RESOURCE RECREATION

Objective: To verify the accuracy of the exported ARM Templates and ensure that they can be used to recreate Azure resources in a disaster scenario.

Steps:

1. Choose a set of Azure resources from the inventory for this drill.
2. Export the ARM Templates for the selected resources using the Azure Portal or Azure CLI.
3. Review the exported ARM Templates for accuracy and completeness, ensuring that they include all necessary resource configurations and dependencies.
4. Create a separate resource group in Azure to deploy the resources for this drill, ensuring that it does not interfere with the existing environment.
5. Deploy the resources using the ARM Templates in the new resource group, either through the Azure Portal, Azure CLI, or another deployment method (e.g., PowerShell, Azure DevOps).
6. Validate that the newly created resources in the separate resource group have the same configurations and dependencies as the original resources.
7. Measure the time taken to complete the deployment and identify any bottlenecks or areas for improvement in the ARM Template deployment process.

5.3 CONTAINER REGISTRY IMAGE RESTORATION AND SECURITY

Objective: Ensure the ability to restore deleted or compromised container images in the Azure Container Registry (ACR) and validate the effectiveness of the continuous deployment pipelines in quickly recovering and deploying secure images.

Steps:

1. Simulate a disaster scenario by either deleting the selected container image from the ACR or introducing a security vulnerability into the image (e.g., altering its contents or modifying its metadata).
2. Trigger the build and push pipelines for the affected container image.
3. Monitor the pipeline execution to ensure that the pipelines are effectively restoring and securing the container image. Verify that the restored image is pushed back to the ACR and that any security vulnerabilities have been addressed.
4. Deploy the restored and secured container image to the appropriate environment.
5. Validate that the restored container image is functioning correctly in the target environment and that no residual security vulnerabilities are present.
6. Measure the time taken to complete the recovery process.

6. RECOMMENDATIONS

In this section we summarize the recommendations to be applied to the EU-CSI infrastructure:

- Setup AKS Backup Extension for having a full backup of all containers used.
- Utilize the native-Azure tools for a full system backup of the Email Server VM with a retention policy of 7 days.