**The Phases of Chaos Engineering:**

A diagram of a process

Description automatically generated

**Chaos Engineering Principles:**

1. Plan an experiment: Create a hypothesis. What could go wrong?
2. Contain the Blast Radius: Execute the smallest test, that will teach you something
3. Scale or Squash: Find an issue? Job well done. Otherwise, increase the blast radius until you are at full scale.

**Few common examples:**

* Simulating failure of a micro-component
* Turning a virtual machine off to see how a dependency reacts.
* Simulating a high CPU or Memory load.
* Disconnecting the system from the data centre.
* Injecting latency between services.
* Randomly causing functions to throw exceptions (also known as function-based chaos).
* Adding instructions to a program and allowing fault injection (also known as code insertion).
* Disrupting syncs between system clocks.
* Emulating I/O errors.
* Causing sudden spikes in traffic.
* Injecting complex failures.

**Chaos Engineering Tooling**

 A black and yellow text

Description automatically generated A purple text on a white background

Description automatically generated

A logo of a cartoon animal

Description automatically generated A green logo with black text

Description automatically generated A logo with a purple and black design

Description automatically generated with medium confidence

A seal logo with text

Description automatically generated

A black text on a white background

Description automatically generated

**Problem Statement:**

* Chaos Mesh is designed to address the challenges of ensuring system resilience and reliability in a dynamic microservices environment.
* Organizations often struggle with unexpected system failures, and it is crucial to understand how systems behave under different failure scenarios.
* The primary problem is to identify weak points and ensure that the system can recover quickly from disruptions without significant downtime or impact on users.

**Solution:**

Chaos Mesh provides a powerful and flexible framework for chaos engineering, allowing users to simulate a variety of failure scenarios within Kubernetes clusters. By inducing controlled disruptions, Chaos Mesh helps teams test the robustness and resilience of their applications. The solution includes:

* Injection of faults like network delays, pod failures, and CPU/memory stress.
* Visualization of experiments and results through a web dashboard.
* Integration with continuous integration/continuous deployment (CI/CD) pipelines.

**Chaos Mesh Overview:**

Chaos Mesh is built on Kubernetes CRD (Custom Resource Definition). To manage different Chaos experiments, Chaos Mesh defines multiple CRD types based on different fault types and implements separate Controllers for different CRD objects.

* **Chaos Dashboard**
* **Chaos Controller Manager**
* **Chaos Daemon**

A diagram of a software application

Description automatically generated

**How Chaos Mesh Works:**

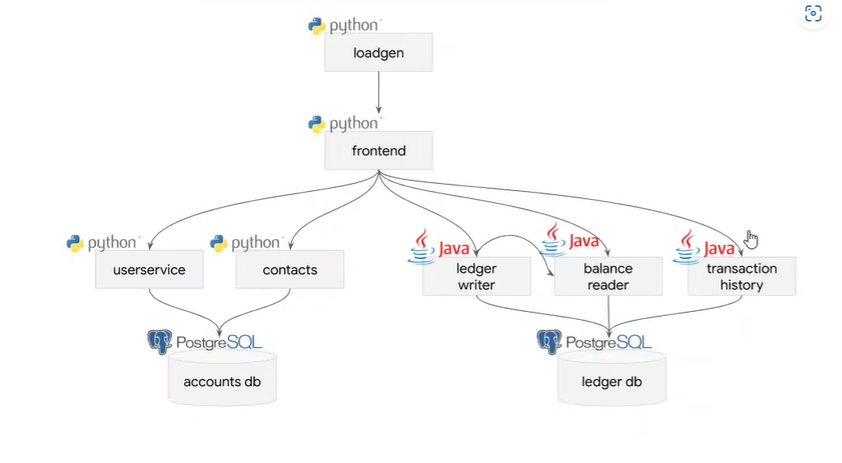
1. **User Interaction:**
   * Users define chaos experiments using Kubernetes custom resource definitions (CRDs)
2. **Controller Manager:**
   * The Controller Manager watches for changes in Chaos Mesh CRDs.
   * Upon detecting a new or updated CRD, it schedules the experiment to specific Chaos Daemon to execute the experiment.
3. **Chaos Daemon Execution:**
   * The Chaos Daemon receives instructions from the Controller Manager.
   * It injects the specified failure into the target pod or node.

**Approach:**

**Steps to install Chaos Mesh:**

Prerequisite:

1. Kubernetes cluster along with a node group.
2. Install helm
3. Bank of Anthos application and database running over the **GKE** **cluster** with an external IP address to access the UI.
4. **Bank of Anthos** Application Architecture.



1. Bank of Anthos application UI

A screenshot of a computer

Description automatically generated

**Install Chaos Mesh:**

1. Create a namespace ***chaos-mesh*** over the Kubernetes cluster.

***kubectl create ns chaos-mesh***

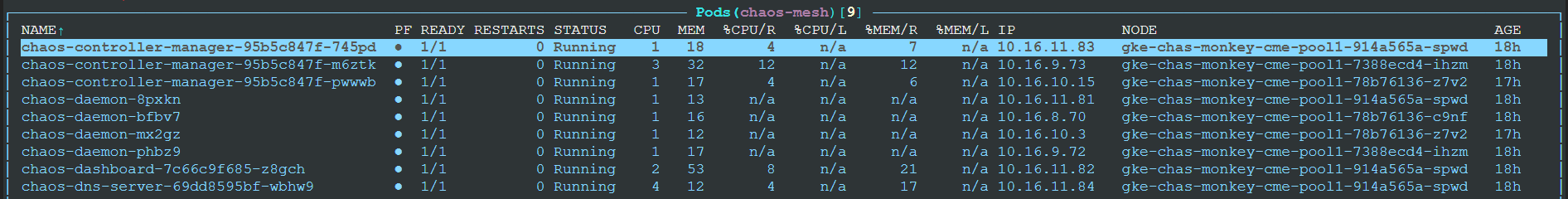
1. Install chaos using helm repo.

***helm repo add chaos-mesh*** [***https://charts.chaos-mesh.org***](https://charts.chaos-mesh.org)

***helm install chaos-mesh chaos-mesh/chaos-mesh -n=chaos-mesh --version 2.7.0***

To test everything is installed & running properly, try this out:

1. Use ***k9s*** to view the Kubernetes cluster. Navigate to ***chaos-mesh*** namespace.
2. Check the resources available in the ***chaos-mesh*** namespace. Refer to the screenshot below.



**Steps to run today’s experiments:**

* **Experiment-1:**

To terminate pod and keep it down for 60 seconds. Experiment configurations are as mentioned below.

A screen shot of a computer

Description automatically generated

* Run the below mentioned command to start the experiment.

***kubectl apply -f pod-kill.yaml -n bank-of-anthos***



* Once the above command is run, a ***Networkchaos*** resource will be created over the mentioned namespace while terminating the target pod and putting the same down for ***60 seconds***.

A screen shot of a computer

Description automatically generated

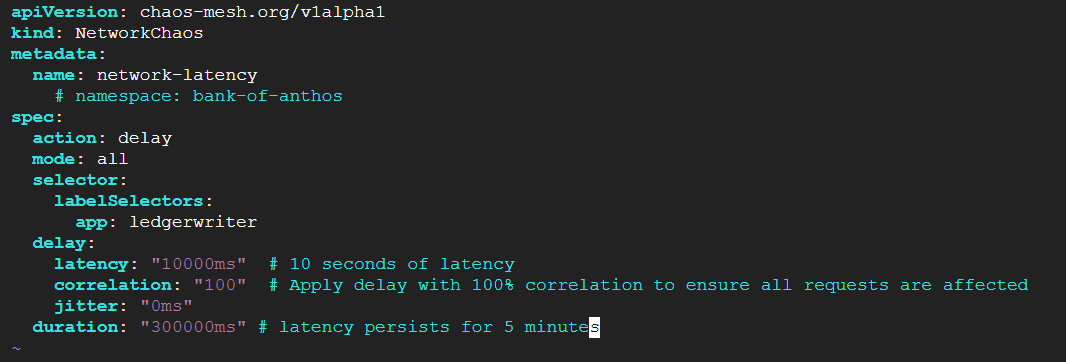
* If we observe in the below screenshot the transaction data doesn’t appear since we have terminated the ***ledger-db*** pod for ***60 seconds***.

A screenshot of a computer screen

Description automatically generated

* **Experiment-2:**

Induce 10 sec latency to a pod for a duration of 5 minutes. Experiment configuration is as mentioned below.

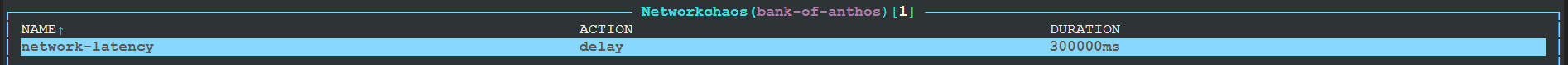


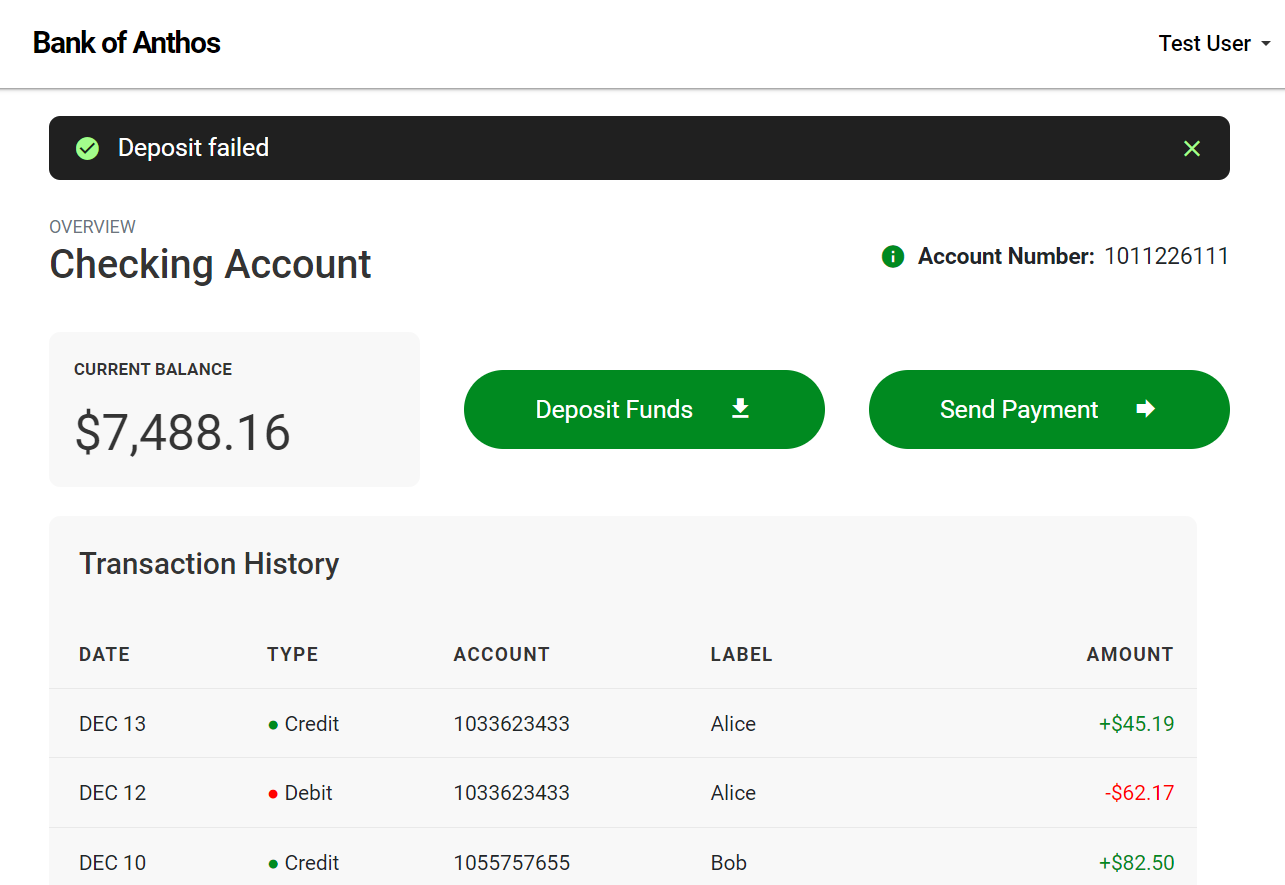
* Run the below mentioned command to start the experiment.

***kubectl apply -f latency.yaml -n bank-of-anthos***



* Once the above command is run, a ***Networkchaos*** resource will be created over the mentioned namespace while inducing **10 *second*** latency to reach the service for ***5*** ***minutes*** duration over the ***Bank of Anthos*** application UI.





* **Experiment-3:** Network partition between ***ledger-db*** & ***ledgerwriter*** MS. Experiment file is as mentioned below.

A screen shot of a computer

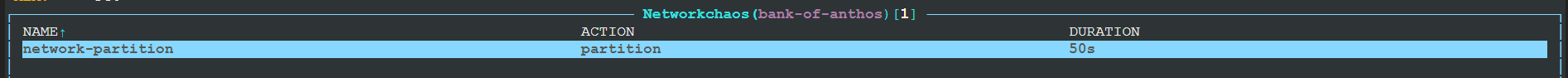
Description automatically generated

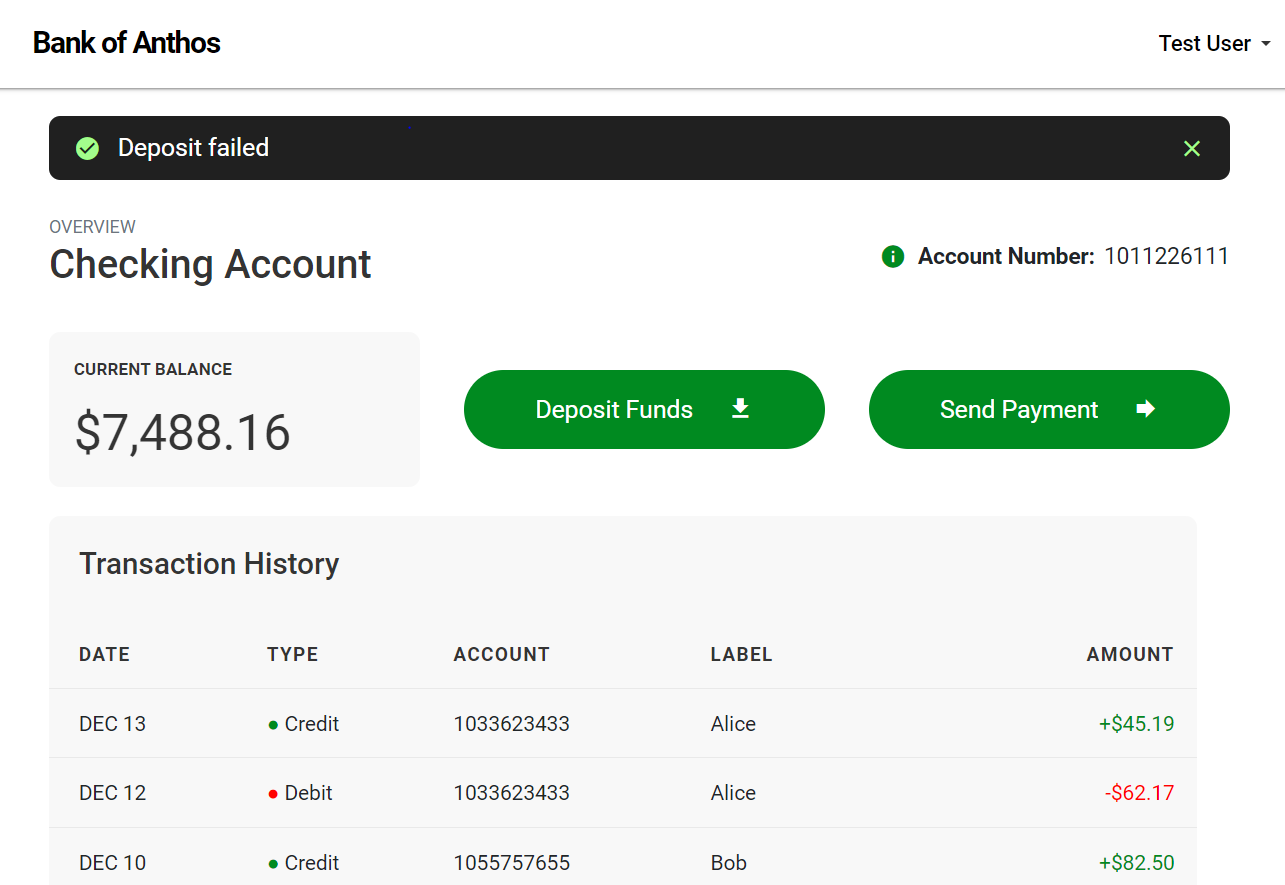
* Run the below mentioned command to start the experiment.

***kubectl apply -f mesh.yaml -n bank-of-anthos***



* Once the above command is run, a ***Networkchaos*** resource will be created over the mentioned namespace while partitioning the network connect between ***ledger-db*** & ***ledgerwriter*** pods in a way that the ***ledgerwriter*** MS can’t perform its functionality like send money & deposit money for a duration of ***50 sec***.





**Results**

Through the application of Chaos Mesh, organizations can achieve the following outcomes:

* **Enhanced Resilience**: Identify and mitigate weak points in the system to ensure higher availability and reliability.
* **Improved Recovery**: Test and validate recovery mechanisms to ensure systems can quickly bounce back from disruptions.
* **Increased Confidence**: Gain confidence in the system's ability to handle real-world failures through continuous and automated chaos testing.
* **Better Preparedness**: Develop a deeper understanding of system behaviour under stress, leading to proactive issue resolution before they impact end-users.

**Suggestion:**

In-order to improve our application resilience enable the below mentioned key features to minimize impact during chaos experiment.

* **Redundancy and Fault Tolerance**
* **Auto-scaling**
* **Circuit Breakers**
* **Health Checks and Readiness Probes**
* **Graceful Degradation**
* **Service Mesh**
* **Observability and Monitoring**

By incorporating these key features into our system's architecture, We can ensure that services remain available and responsive even during chaos experiments. The system should be able to:

* Handle increased latency or failures gracefully.
* Automatically recover and scale to meet demand.
* Maintain core functionality and provide seamless user experiences.

By incorporating Chaos Mesh into your testing strategy, you can build more robust and resilient systems capable of withstanding unexpected challenges.

Thank You ..!!