Abukhassym Khydyrbayev

SE-2318

Introduction to SRE
(Site Reliability Engineer)

Assignment_1

SRE Assignment Report: Setting Up an SRE Environment and Monitoring Tools

1. Introduction

This report documents the comprehensive setup of a Site Reliability Engineering (SRE) environment, focusing on containerization, networking, storage, backup mechanisms, and essential monitoring tools. The assignment covers key aspects of modern infrastructure management using Docker and various SRE tools.

2. Docker Installation and Configuration

2.1 Docker Platform Selection

Containerization Approach:
Docker is a lightweight containerization platform that allows running applications in isolated environments. Unlike traditional virtualization, Docker shares the host OS kernel, providing:

- Improved efficiency
- Lightweight container deployment
- Consistent environments across different systems

Platform Options:

- **Docker Desktop**: Recommended for local development
- **Docker Engine**: Ideal for server and production environments

2.2 Installation Process

Operating System-Specific Installation

1. **Windows**
    - Download Docker Desktop for Windows
    - Ensure WSL2 (Windows Subsystem for Linux) is enabled for Windows 10 Home
2. **macOS**
    - Download Docker Desktop for Mac
    - Supports both Intel and Apple Silicon processors
3. **Linux**
    - Follow distribution-specific instructions
    - Detailed installation guide available at: https://docs.docker.com/engine/install/

Verification

```
docker --version
```

Initial Container Test

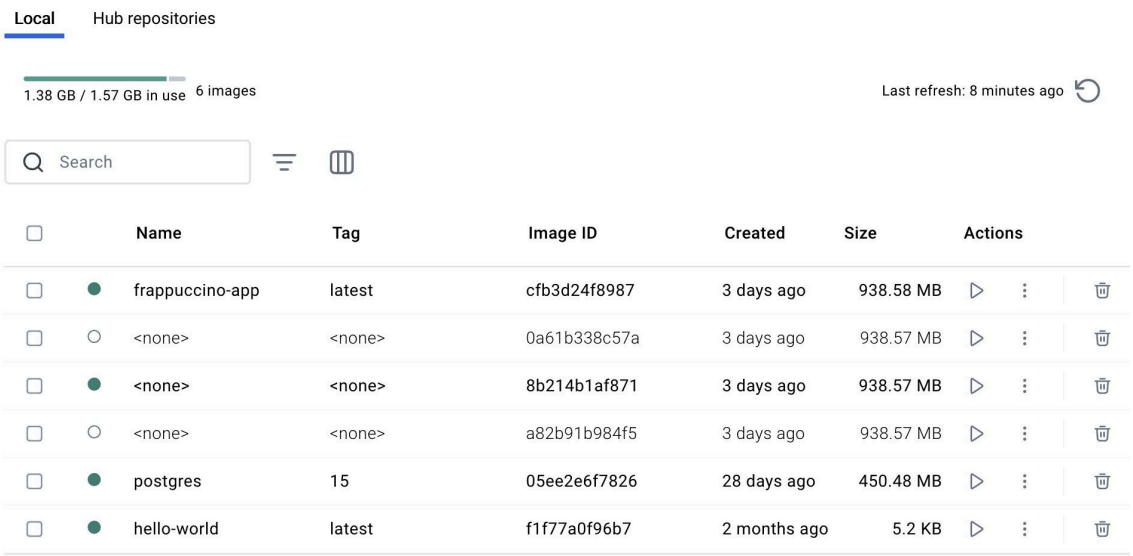
```
docker run hello-world
```

```
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker --version
Docker version 28.0.1, build 068a01e
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c9c5fd25a1bd: Pull complete
Digest: sha256:7e1a4e2d11e2ac7a8c3f768d4166c2defeb09d2a750b010412b6ea13de1efb19
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (arm64v8)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Expected Output:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

		Name	Container ID	Image	Port(s)	CPU (%)	Last sta	Actions
		angry_gagarin	3a7360527302	8b214b1af		N/A		
		unruffled_blackb	30c1c0756617	hello-world		N/A	5 second	
	>	frappuccino	-	-	-	N/A	3 days ag	



2.3 Network Configuration

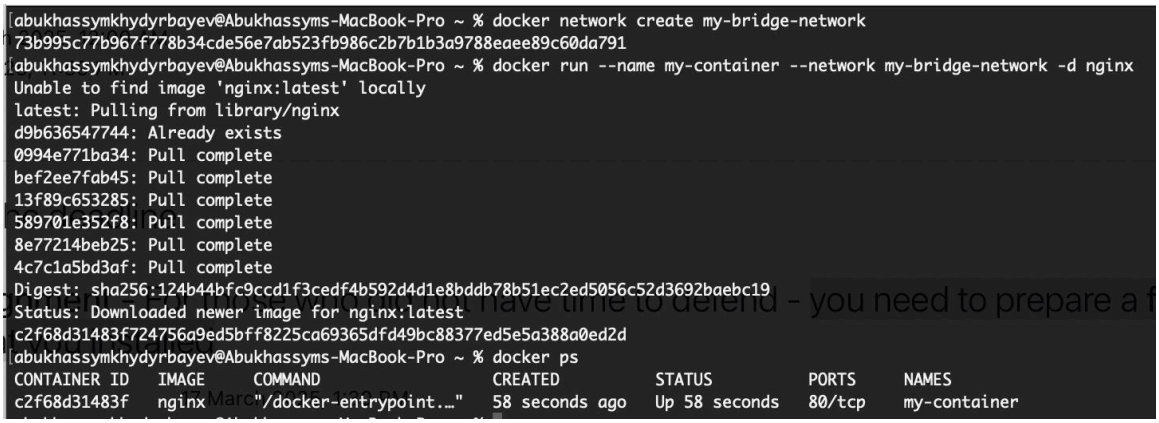
Docker provides multiple networking modes:

4. **Bridge Network**
    - Default network mode
    - Allows containers to communicate with each other and the host
5. **Host Network**
    - Containers share the host's network stack
    - Direct network access
6. **Overlay Network**
    - Used for multi-host communication
    - Ideal for Docker Swarm or Kubernetes environments
7. **Macvlan Network**
    - Assigns a MAC address to a container
    - Makes container appear as a physical device

Network Configuration Example:

```
# Create a custom bridge network
docker network create my-bridge-network

# Run a container on the custom network
docker run --name my-container --network my-bridge-network -d nginx
```

```
# Verify container running
docker ps
```



2.4 Storage Configuration

Docker supports three storage types:

8. **Volumes**
   - Persistent storage outside container lifecycle
   - Managed by Docker
9. **Bind Mounts**
   - Maps a host directory to a container directory
   - Provides direct host filesystem access
10. **tmpfs Mounts**
    - Temporary storage inside the container
    - Stored in host system memory

Storage Configuration Example:

```
# Create a volume
docker volume create my-volume

# Run container with volume mounted
docker run --name my-container -v my-volume:/data -d nginx

# Verify volume mounting
docker inspect my-container
```

```
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker volume create my-volume
my-volume
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker run --name my-container -v my-volume:/data -d nginx
b9301775763dc108f3e5514d8a84d3229d8e571cd6447652fcfd1faf1bd2bb48
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker inspect my-container
```

2.5 Backup Mechanisms

Two primary backup methods:

11. **Commit Method**

```
docker commit my-container my-backup-image
```

12. **Docker Save/Load Method**

```
# Export image
docker save -o my-backup.tar my-backup-image

# Import image
docker load -i my-backup.tar
```

```
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker commit my-container my-backup-image
sha256:7e37e732a6ee748c8d74541d8ec9bde7921f757fd55b77699e4023700eddb378
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker save -o my-backup.tar my-backup-image
docker load -i my-backup.tar
Loaded image: my-backup-image:latest
```

3. SRE Monitoring Tools

3.1 Prometheus

Purpose: Monitoring and metrics collection tool

Docker Installation:

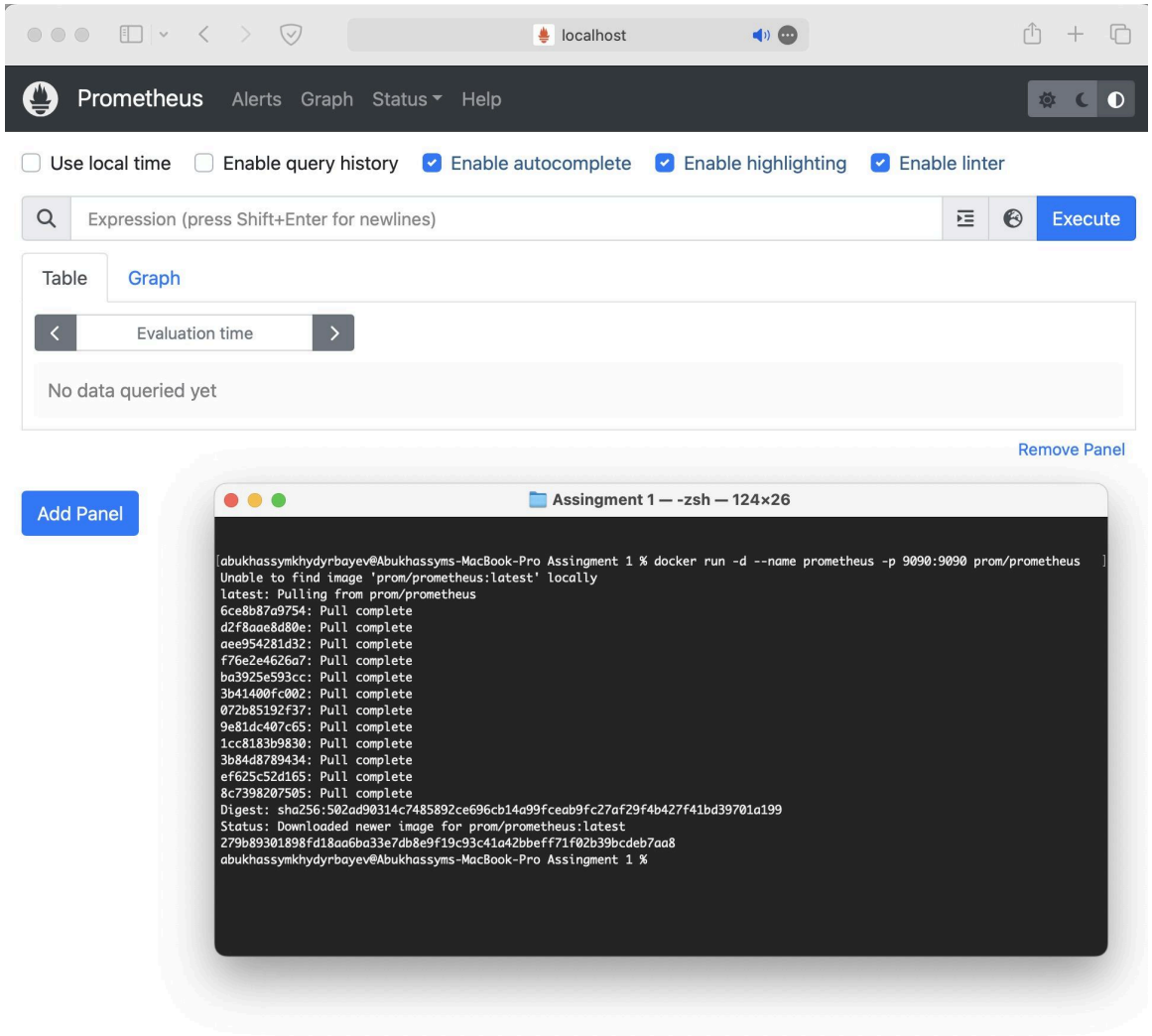
```
docker run -d --name prometheus -p 9090:9090 prom/prometheus
```

Configuration (`prometheus.yml`):

```
scrape_configs:
  - job_name: "prometheus"
    static_configs:
      - targets: ["localhost:9090", "localhost:4000"]
    scrape_interval: 15s
    scrape_timeout: 10s
```

Metric Querying:

```
# Check target health
curl -X GET "http://localhost:9090/api/v1/query?query=up"

# Detect HTTP 500 errors
curl -X GET
"http://localhost:9090/api/v1/query?query=sum(rate(http_requests_total{status=
~"5.."}[5m]))"
```



3.2 Alertmanager

Purpose: Handle and route alerts from Prometheus

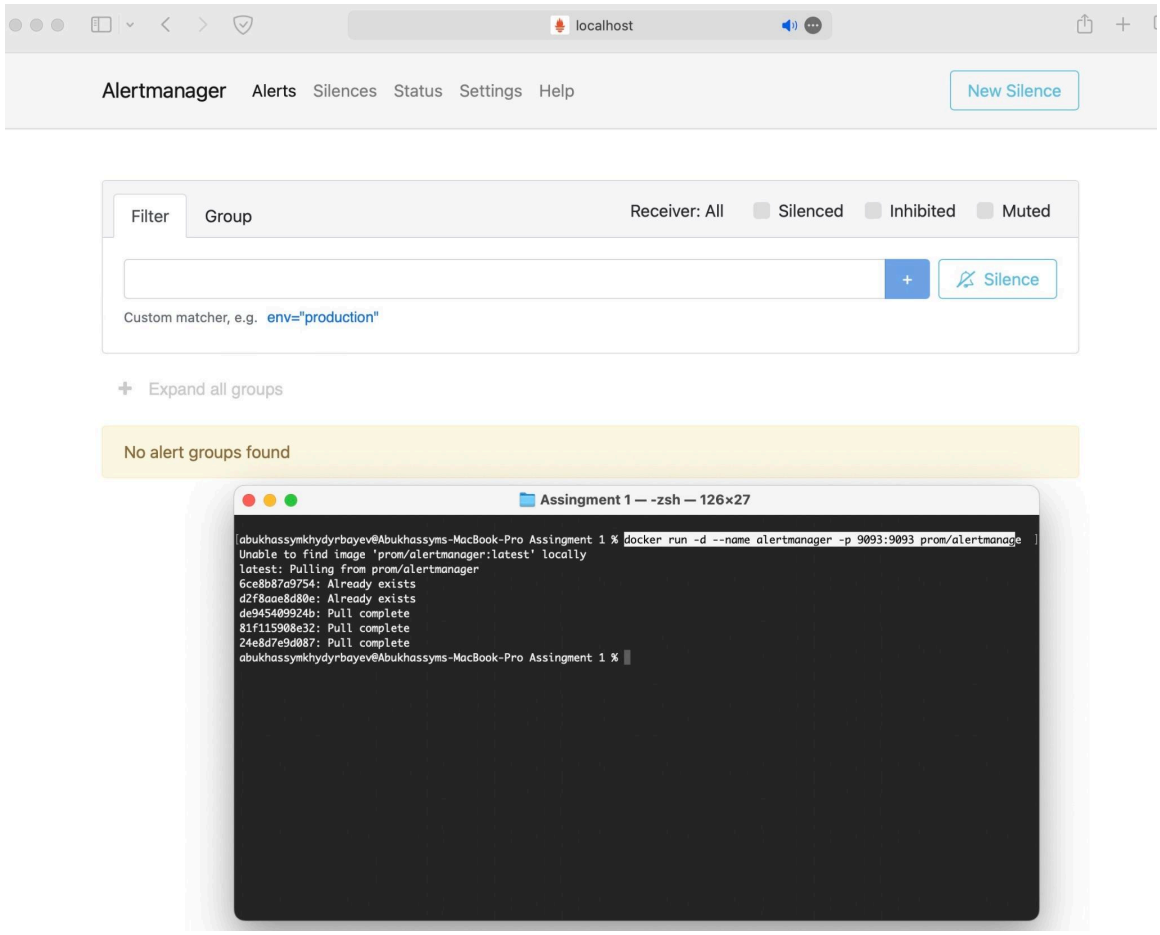
Docker Installation:

```
docker run -d --name alertmanager -p 9093:9093 prom/alertmanager
```

Sample Alert Configuration:

```
route:
  receiver: 'team-email'
  group_by: ['severity']
  routes:
    - match:
        severity: 'critical'
      receiver: 'pagerduty'

receivers:
```

```
  - name: 'team-email'
    email_configs:
      - to: 'team@example.com'
  - name: 'pagerduty'
    pagerduty_configs:
      - service_key: '<your-pagerduty-key>'
```



3.3 ELK Stack

Purpose: Log management and analysis

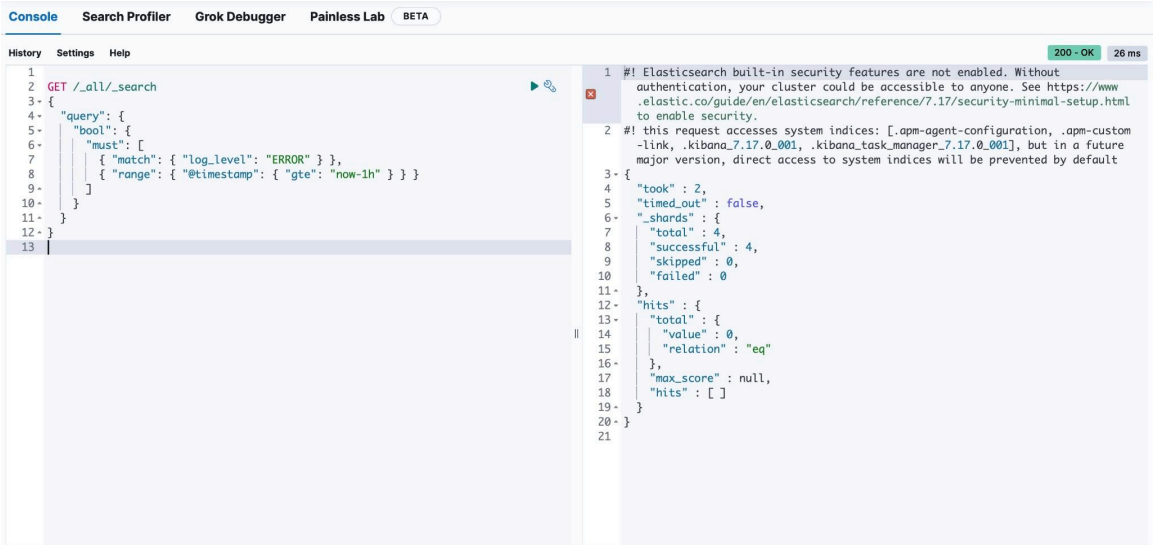
Docker Installation:

```
# Elasticsearch
docker run -d --name elasticsearch -p 9200:9200 -p 9300:9300 -e
"discovery.type=single-node"
docker.elastic.co/elasticsearch/elasticsearch:7.17.0

# Kibana
docker run -d --name kibana -p 5601:5601 --link elasticsearch:elasticsearch
docker.elastic.co/kibana/kibana:7.17.0
```

Log Search Examples:

```
# Search for Errors
GET /application-logs/_search
{
  "query": {
    "match": {
      "log_level": "ERROR"
    }
  }
}

# Correlate Logs
GET /_all/_search
{
  "query": {
    "bool": {
      "must": [
        { "match": { "log_level": "ERROR" } },
        { "range": { "@timestamp": { "gte": "now-1h" } } }
      ]
    }
  }
}
```

```
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker run -d --name grafana -p 3000:3000 grafana/grafana

Unable to find image 'grafana/grafana:latest' locally
latest: Pulling from grafana/grafana
6e771e15690e: Pull complete
3dd8d8d6a974: Pull complete
b23536fc5777: Pull complete
3a118e96ded1: Pull complete
8318ded4e9cc: Pull complete
756b878b6aa7: Pull complete
de210d0897ff: Pull complete
bf988e0055d8: Pull complete
1cb82048f91d: Pull complete
babadc1b811e: Pull complete
Digest: sha256:62d2b9d20a19714ebfe48d1bb405086081bc602aa053e28cf6d73c7537640dfb
Status: Downloaded newer image for grafana/grafana:latest
3850daa0016d123e3cbf97afb3ecca03300826e46b5cce7657c2f4172df1b07b
abukhassymkhydyrbayev@Abukhassyms-MacBook-Pro ~ % docker ps -a

CONTAINER ID   IMAGE                                                  COMMAND                  CREATED              STATUS
3850daa0016d   grafana/grafana                                        "/run.sh"                About a minute ago   Up About a
06c26b19c51d   docker.elastic.co/kibana/kibana:7.17.0                 "/bin/tini -- /usr/l…"   5 minutes ago        Up 5 minut
af93142940ac   docker.elastic.co/elasticsearch/elasticsearch:7.17.0   "/bin/tini -- /usr/l…"   5 minutes ago        Up 5 minut
30c1c0756617   hello-world                                            "/hello"                 21 hours ago         Exited (0)
```



3.4 Grafana

Purpose: Visualization of metrics and monitoring data

Docker Installation:

```
docker run -d --name grafana -p 3000:3000 grafana/grafana
```

4. Conclusion

This assignment demonstrated:

- Docker installation and configuration
- Network and storage management
- Backup strategies
- Implementation of monitoring tools
- Log management and visualization