

Financial Research Involving Data Analysis (FRIDA)

GOALS

In order to be able to scale our platform we need to asure that one core part of the architecture is reliable and scalable for our purposes, specifically the message broker, Apache Kafka in this case.

O1. Measure and compare the performance
 of the message broker and network while scaling
 up the volume of messages processed

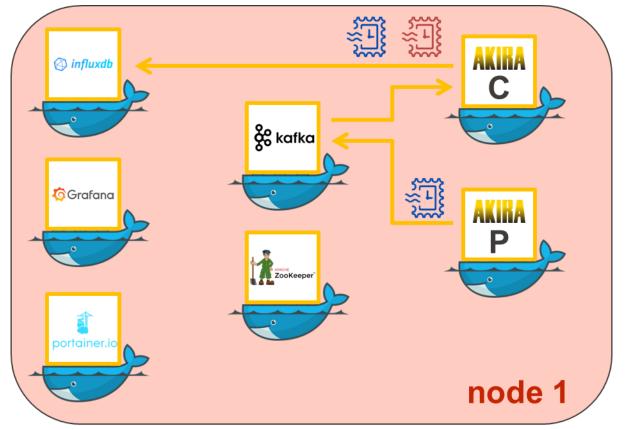
CHALLENGES

Several challenges were faced:

- Learn how to use Grid5000 and its tool
- Get familiarized with Docker Swarm
- Improve NTP accuracy by using PTPd
- Obtain a suitable approach to reproduce the experiment

DEMO SETUP

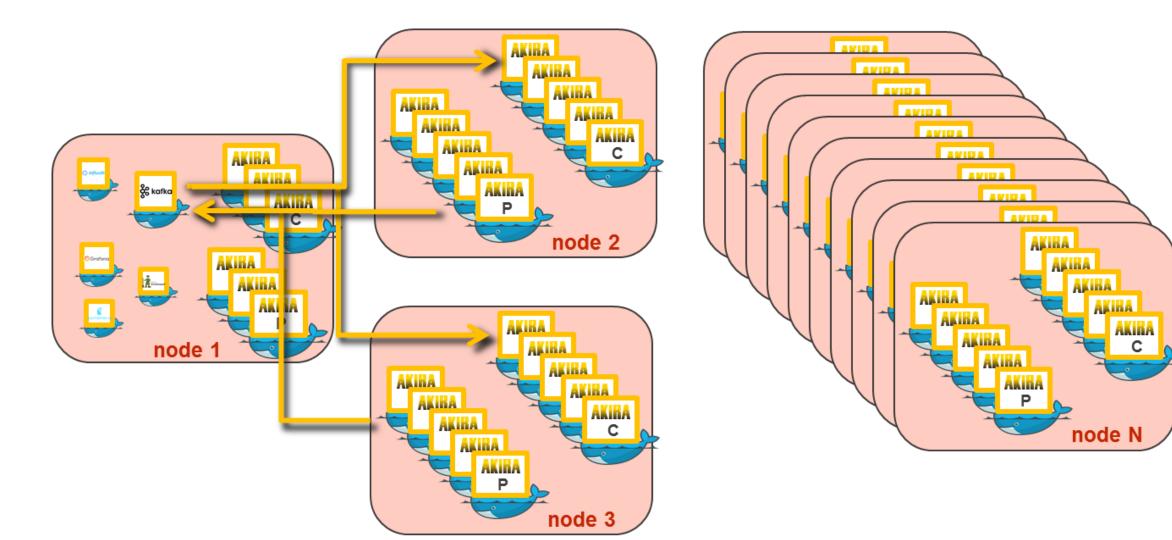




1. Deploy the stack in the swarm

2.Add more nodes

3. Scale containers / services



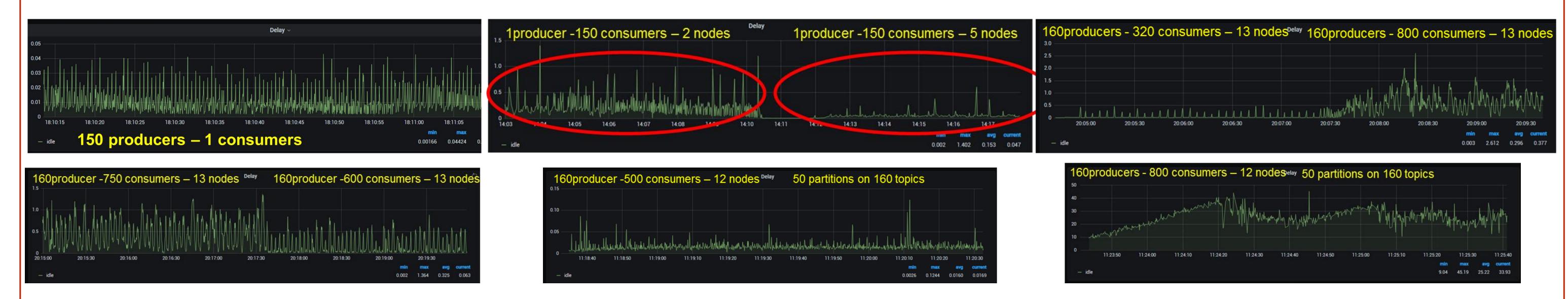
To operate the experiment and get the results, a combination of several tools was used, as Portainer, Influxdb and Grafana

portainer.io	=	JEIVICES			My account of tog out	6 册 Host Overview	مہ ید		④ Last 15 minutes → Q 및 5s →
Home	*	Services			🌣 Settings		مہ بر 		
₩ PRIMARY Dashboard @		C Update 📋 Remove + Add	service			datasource InfluxDB - host Iocalhost - Q			
App Templates		Q Search				Delay			
Stacks Services	=	□ > Name ↓ [*]	Stack Image	Scheduling Mode	Put	+ 1.00			
Containers		> portainer_agent	portainer portainer/agent:latest	global 2 / 2	-	0.75			
Images Networks	e .**	> portainer_cadvisor	portainer gcr.io/google-containers/cadvisor:late	st replicated 1 / 1 \$Scale	2 7	0.50			
Volumes	&	> portainer_collectd	portainer puckel/docker-collectd:latest	global 2 / 2	-	<u>۵.25</u>			
Configs		portainer_consumer	portainer teiichi/consumer:latest	replicated 1 / 100 100	¢ ×⊠ ⁻	0 <u>////////////////////////////////////</u>		14:23 14:24 14:25 14:26 14:27	14:28 14:29 14:30 14:31
Secrets Swarm	â	portainer_grafana	portainer grafana/grafana:latest	replicated 1 / 1 \$Scale	ピ 3	🗘 — idle			min max avg current 0.007 0.873 0.104 0.064
SETTINGS		portainer_influxdb	portainer influxdb:latest	replicated 1 / 1 \$ Scale	1 2 8				
Extensions Users	₽	> portainer_kafka	portainer kafka-docker_kafka:latest	replicated 1 / 1 \$ Scale	2 9	100	Memory		I Usage 🗸
Endpoints	÷	> portainer_kafka_manager	portainer hlebalbau/kafka-manager:stable	replicated 1 / 1 \$ Scale	2 7	1.0 B		1.0	min max avg
Registries	8	> portainer_portainer	portainer portainer/portainer:latest	replicated 1 / 1 \$Scale	2 8	0.5 B	No data	0.5 No deta	
Settings	\$	> portainer_producer	portainer teiichi/producer:latest	replicated 150 / 150 \$Scale	-	0 B		No data	
portainer.io 1.2	24.1	<		Items per page 10 🗸	• • • •	-0.5 B		-0.5	

To handle the nodesin Grid5000, some scripting based on OAR was required

malvarez@fsophia:~\$./register_and_get_ip.sh "./worker_node.sh
docker_token_1598537715209158862 172.16.132.9
Reserving node..
Reserved with IP: 172.16.132.26

MORE RESULTS



With the experiment setup shpwn, it has been possible to test different escenarios for different configurations, varying the number of producers, consumers and the number of available nodes. Graphs show the delay time of a message from a producer to a consumer.

 Docker swarm fits perfectly for managing the scalability of our platform

CONCLUSIONS

- Kafka is a very fast and reliable server for our purpose
- Increasing the number of producers does not affect the performance of a server.
- For a single Kafka server, 500-600 producers are the limit for a proper delivery time.
- Fine tunning of the server is required to increase performance and a Kafka servers cluster maybe required.

 Having such amount of resources ready to be used is the most appreciated value of Fed4FIRE. Also the good documentation to get hands on quickly is a plus.

POST MORTEM

- In Stage 2 we will take benefit of different testbeds and will be able to compare different philosophies as in Tengu, GPULab and Grid5000.
- Tesiting Kafka clusters will also be conducted on Grid5000.