



On-Premises Deployment

Addressing the challenges

Mojaloop PI-20 event, October 2022 - David Fry

Who are we?



Miguel

*Mojaloop Principal Architect,
Infitx*



David Fry

*Director - Cloud Infra,
Infitx*



Steve Bristow

*Hybrid-Cloud infrastructure
specialist*

On-Premises Deployment



Majority of Mojaloop deployments have been cloud based

- ✓ Rapidly Scalable
- ✓ Low 'door price' – cost-of-entry
- ✓ Implicit Security

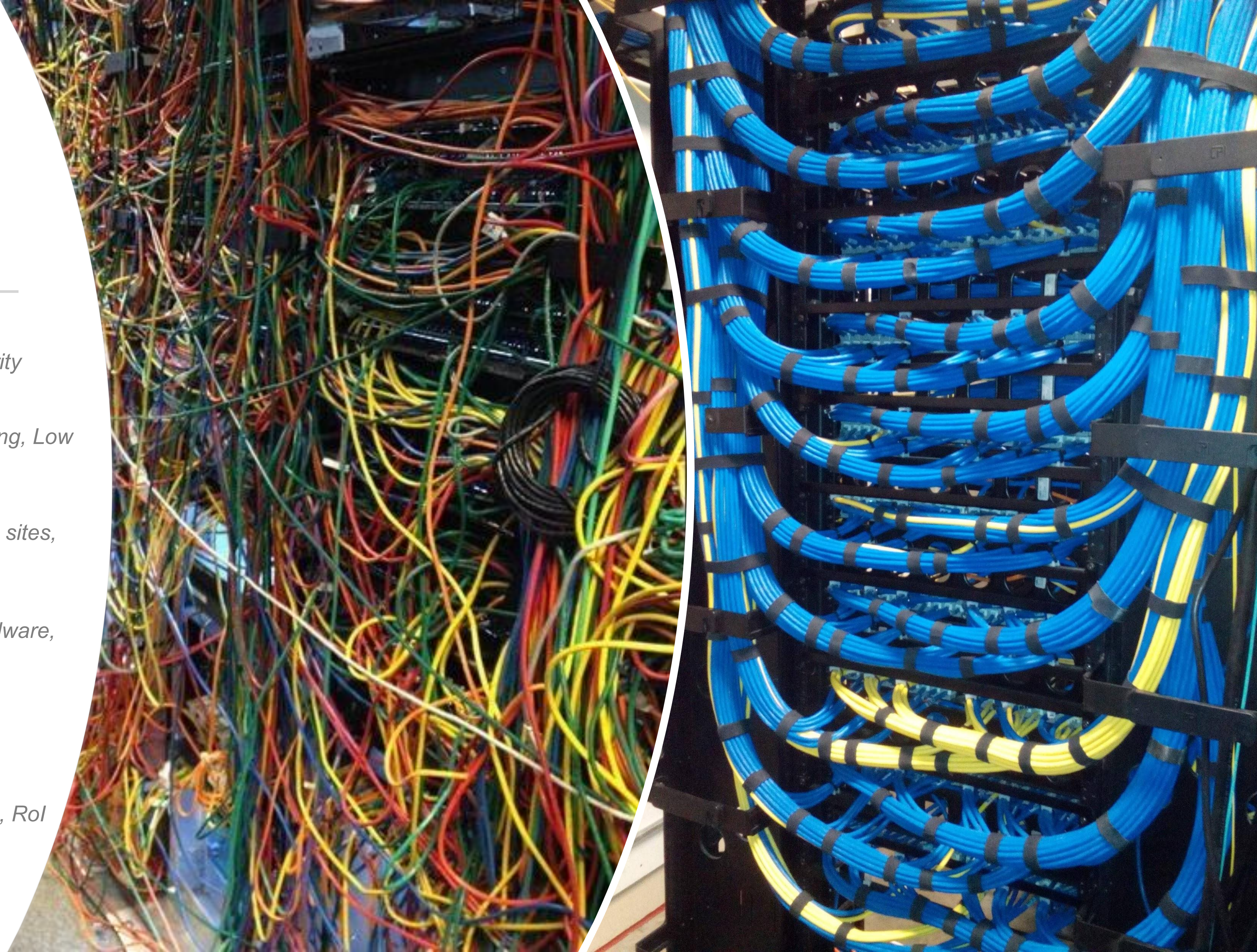
Many Mojaloop customers favour on-premises deployment

- ✓ Transparency – no hidden cloud-limits
- ✓ Visibility – the ability to measure every single facet of a system
- ✓ Resolute control – ownership of problems and security from bottom-to-top
- ✓ Compliance – Clear Data Sovereignty, Legal boundaries and responsibilities.
- ✓ Latency – workloads can be located as close to consumers as possible

- Mojaloop should be available to everyone – but is currently excluded where compliance demands data is stored in-country, and cloud providers do not have presence.
- Sizing a larger mojaloop deployment is not a turnkey process – and mistakes are expensive!
- mojaloop-laC facilitates a rapid, repeatable, standardised deployment on-prem and in-cloud.

The on-prem challenge

- **Appropriate Datacenter(s)**
Cooling, Cabling, Power, Security
- **Adequate connectivity**
Internet Adjacent, Diverse routing, Low Latency
- **Managing Availability**
Maintenance windows, multiple sites, unplanned outages
- **Scaling**
Support contracts, Finding hardware, purchase cost
- **Scaling!**
Quantities, Shipping, BoM
- **Scaling!!**
Predict Growth, Business Case, RoI





Availability

Understanding the cost implications of an availability model

Understanding Availability



	Description	Public Perception	Regulator Perception
1 minutes	Absorbed by DFSP queues.	Unnoticed.	Unnoticed.
5 minutes	Timeout and Retry thresholds reached.	Transactions aborted. Customers Retry.	Unconcerned.
10 minutes	Customer Retries fail.	System is perceived as “Down”.	Interested.
20 minutes	Repeated retries fail.	Reputational Damage.	Report expected.
1+ hours	Outrage cycle begins.	Severe Reputational Damage.	Audit likely.

- Availability is an expensive.
- Risk appetite should be decided, documented and agreed and communicated before Go-Live.
- Solution design should always consider risk-appetite.

Understanding Availability

Is N+1 enough: Active + Passive

Under Normal Operation

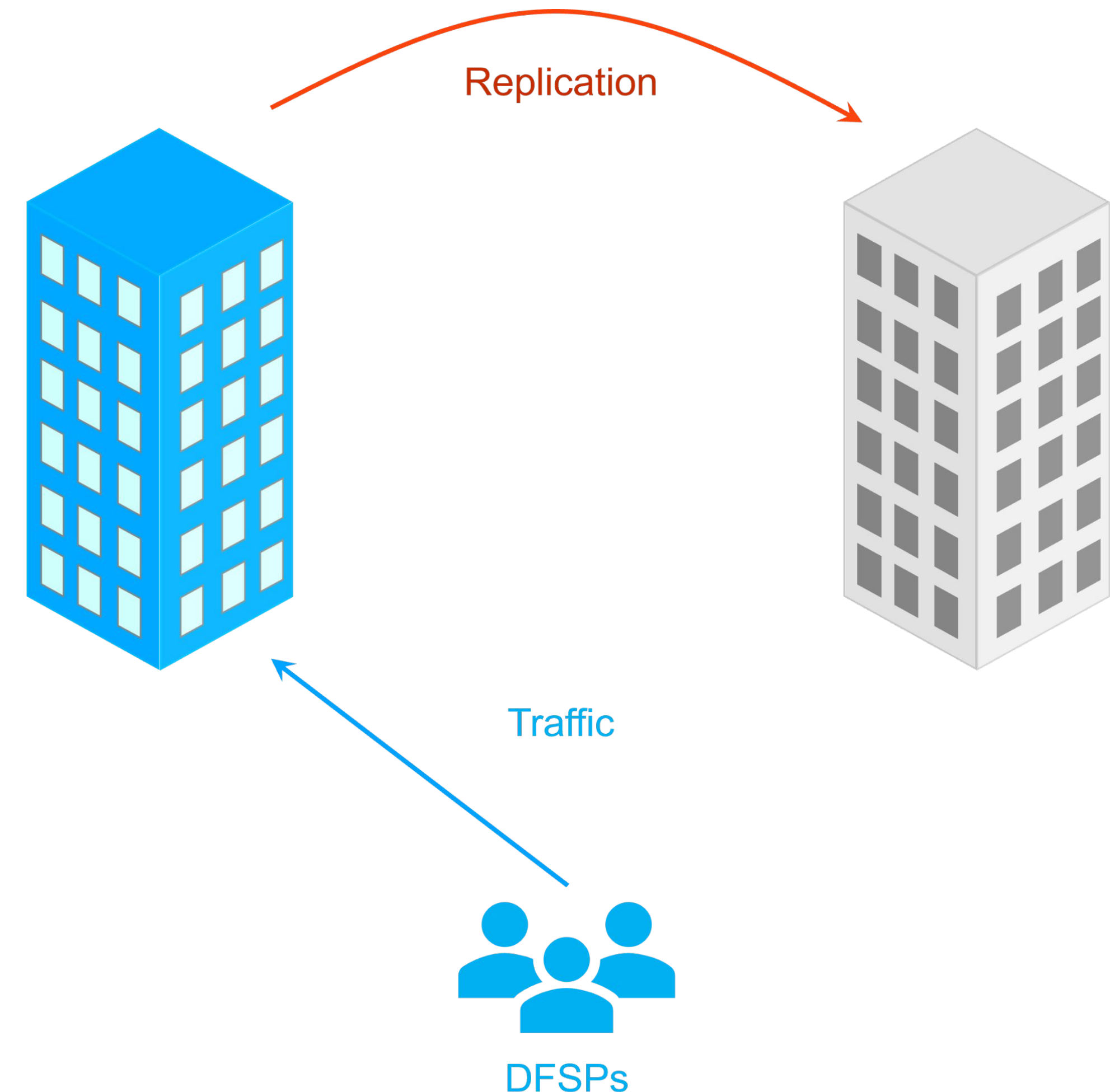
- Every transaction must wait for replication (failures on 'passive' side may still impact 'active' side).
- Maintenance requires pausing replication.
- 50% of infrastructure is available, but not processing live transactions.
- Failover must be tested often.

During a controlled failover

- Interruption to accepting new transactions.
- In-Flight transactions will finish.
- Must be performed during periods of low-throughput.

During an uncontrolled failover

- Complete outage during uncontrolled failover process.
- Most In-Flight transactions will be lost.
- Likely to require **Manual** Failover (See 'Active-Active Split-Brain').



**Not
Recommended**

Understanding Availability

Enough: Active + Active

Under Normal Operation

- Every transaction must wait for replication (issues in one DC may still affect the other).
- Maintenance requires pausing replication **and** incoming traffic to target DC.
- 50% of each DC must be reserved for peak transaction flow.
- Failover must be tested often.

During a controlled failover

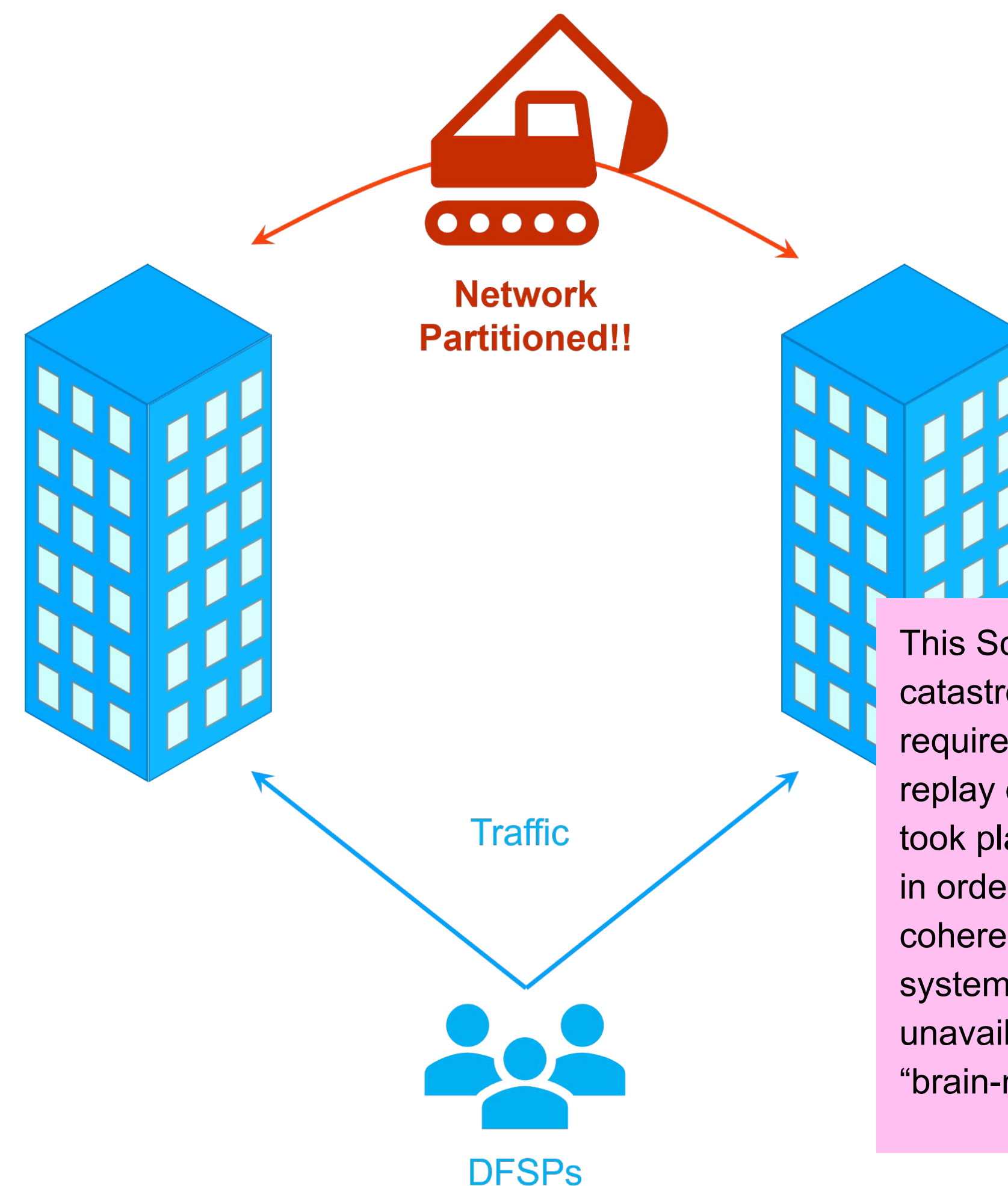
- No interruption to transaction traffic – DC ‘drains’ inbound connections.
- In-Flight transactions will finish.
- Best performed during periods of low-throughput.

During an uncontrolled failover

- In-Flight transactions to failed DC will be lost.
- In-Flight transactions to healthy DC will pause.
- Likely to require **Manual Failover**.
- **Significant Risk of ‘Active-Active Split-Brain’.**

During a Network Partition:

- **Both sites believe the other has failed.**
- Both sites will assume they own the master data set.
- Both sites will continue to transact using independent copies of the database.



This Scenario is generally catastrophic. Recovery requires the chronological replay of all transactions that took place since the partition in order to arrive at a fully coherent database. The system is usually unavailable whilst this “brain-merge” takes place.

Understanding Availability

Is N+1 enough: Active + Active + Active?

Under Normal Operation

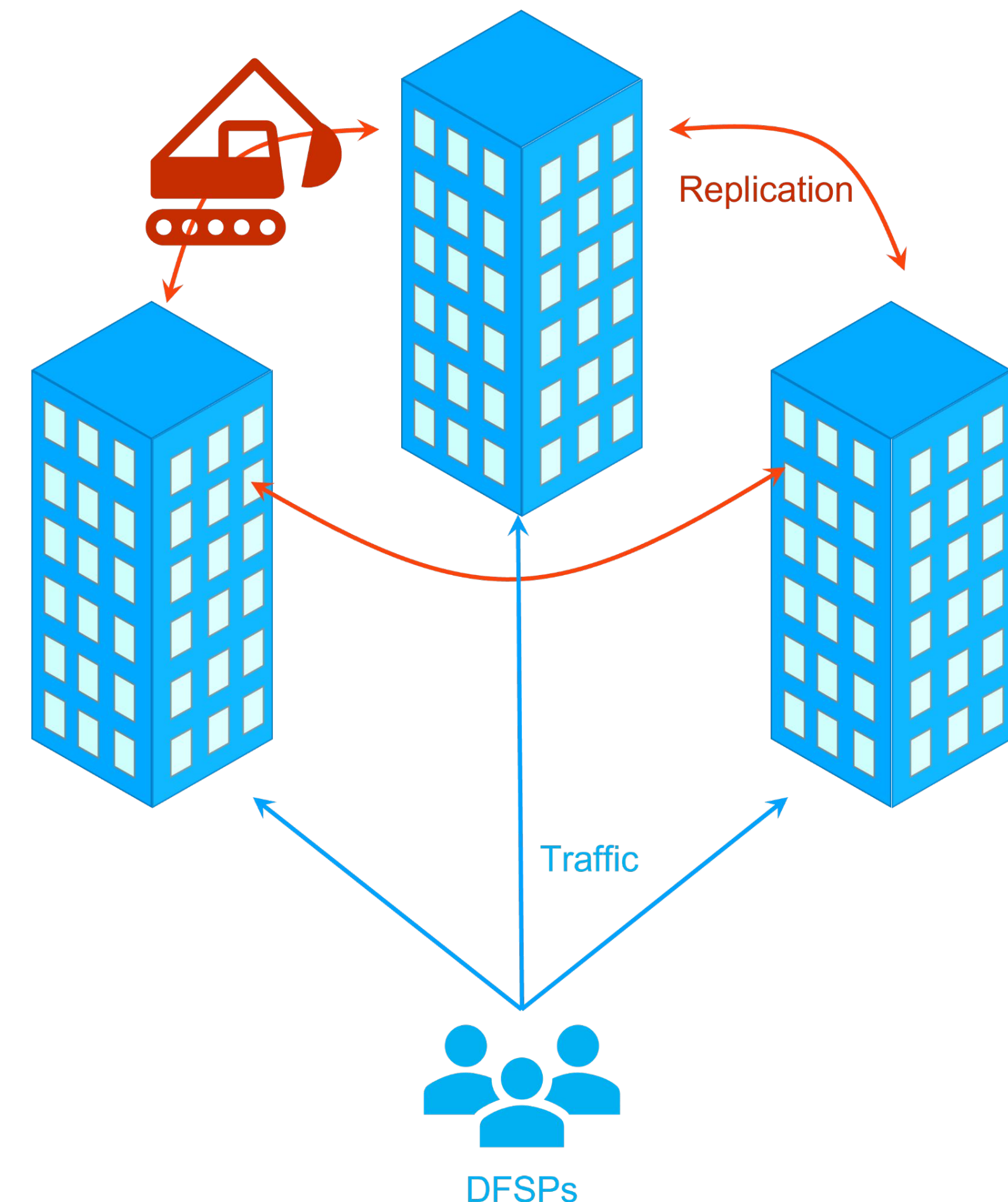
- Every transaction must wait for replication to at least one DC.
- A single DC can be 'drained' for maintenance, leaving an active+active configuration.
- Only 30% of each DC must be reserved for peak transaction flow.
- Failover testing still leaves two operational DCs.

During a controlled failover

- No interruption to transaction traffic – DC 'drains' inbound connections.
- In-Flight transactions will finish.
- Best performed during periods of low-throughput.

During an uncontrolled failover

- In-Flight transactions to failed DC will be lost.
- In-Flight transactions to healthy DC will continue.
- Failover can be fully automatic (however fallback should remain manual).
- Risk of split-brain is significantly minimised.





Scaling

The technical implications of scale

Understanding Scale

The Challenges

Your switch will have different traffic patterns to everyone else's.

The only way is 'try-before-you-buy'.

Nation-scale-growth is unpredictable!

The platform must be easy to scale.

Buying the hardware can be a huge upfront investment!

The platform should grow with your workload and revenue.



There is minimal publicised data available about the real-world performance of mojaloop across different deployments and markets.

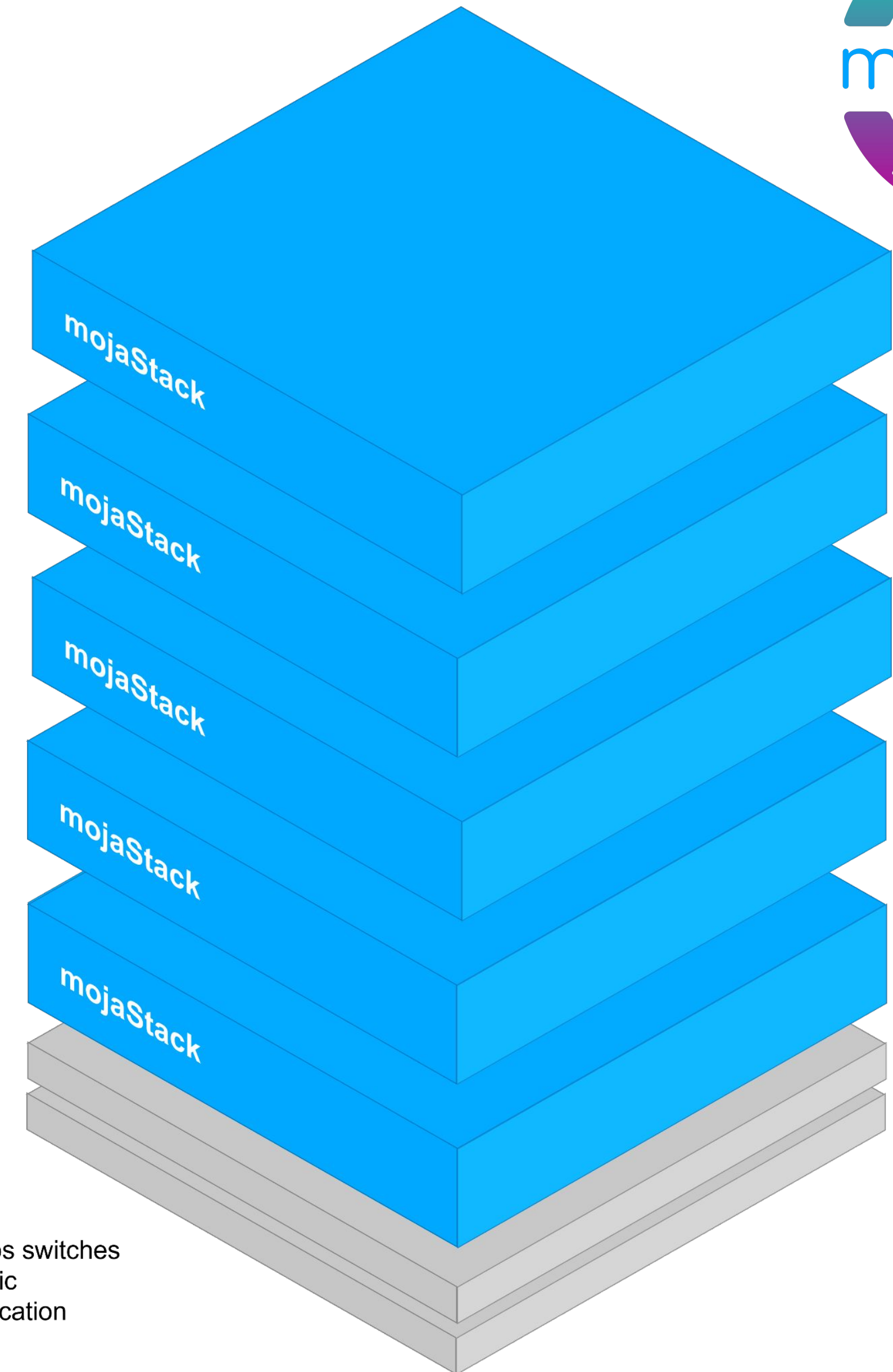
mojaStack



Meet the mojaStack cluster:

- Dual 32 port 100Gbs switches support up to 14 nodes in a stack
- Storage is replicated between nodes – in the event of a failure, workloads move to nearest copy of data.
- 2x100Gbs storage networking ensures sufficient replication bandwidth
- 2x100Gbs mojaloop networking for failover and minimal latency
- ALL services run within the stack – no additional hardware is required (*note: mass archival storage is not included*)
- Open-Source software means no vendor lock-in
- Centralised management (Rook, Kubernetes)
- Commodity Hardware – vendor agnostic
- Cattle - Not Pets!

But what about failures?



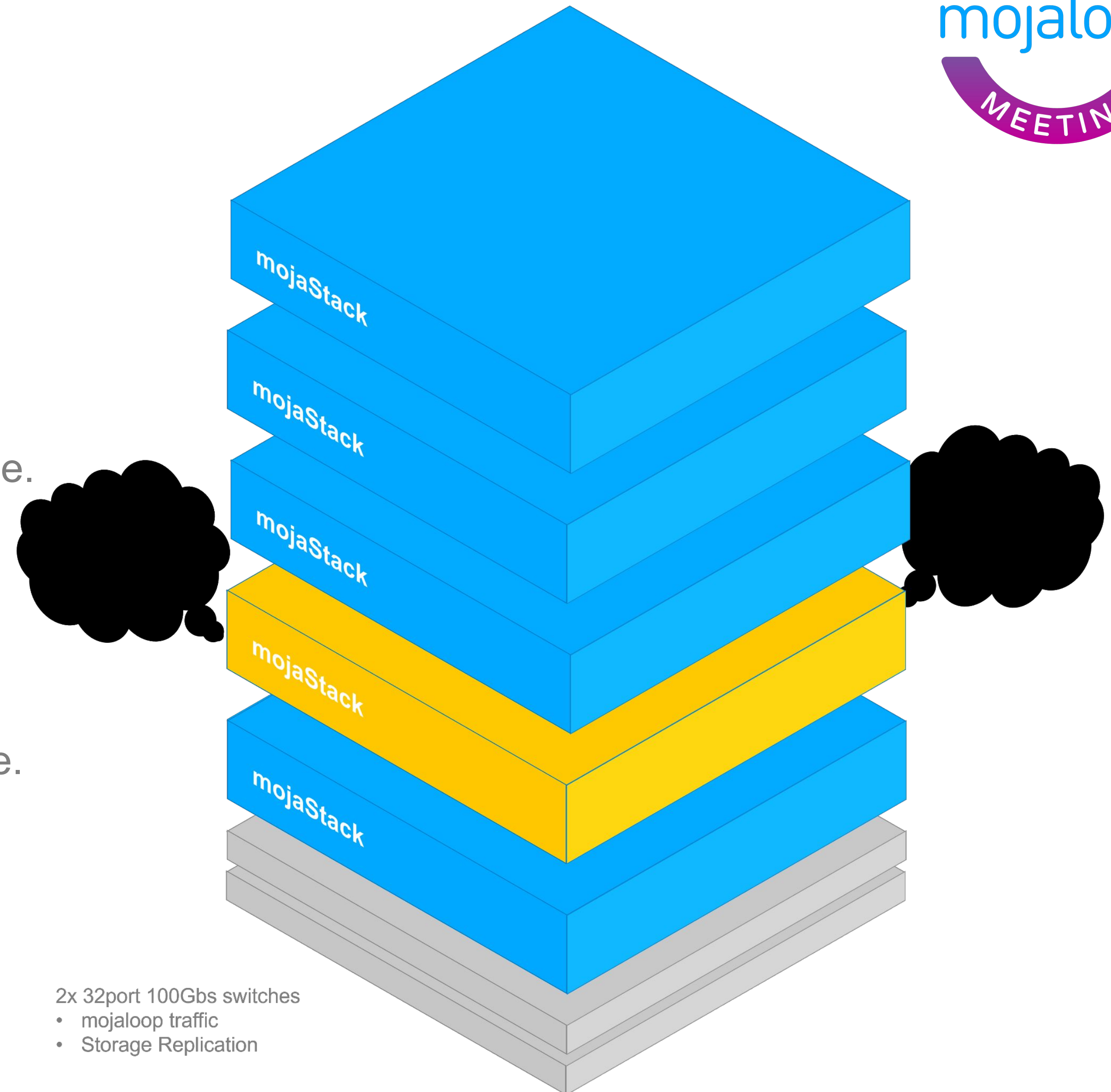
32port 100Gbs switches
mojaloop traffic
Storage Replication

mojaStack



But what about failures?

- In the event of a component failure
 - Node will announce it is degraded.
 - Storage will be mastered on other nodes.
 - Workloads will be re-hosted adjacent to storage.
- In the event of a node failure
 - Node will disappear.
 - Active workloads will fail.
 - Active transactions will fail.
 - Storage will be mastered on other nodes.
 - Workloads will be restarted adjacent to storage.



Understanding Scale - Approach



STEP 1:

Validate baseline workload assumptions






Principles:

- Use baremetal cloud services against a variety of patterns to validate workload characteristic hypothesis.
- Identify optimal price-performance configurations
- Ignore availability concerns

Understanding Scale - Approach



Software stack will consist of:

	microK8S-ha	This kubernetes distro supports easy adding/removal of nodes and HA configurations.
 MetalLB	MetalLB	Load-balancer implementation designed for bare-metal deployment.
 CoreDNS	CoreDNS suite	CoreDNS plugins such as k8s-external/k8s-gateway.
 ceph	Ceph	Ceph storage is super-fast, extremely resilient and highly configurable as block, file or object.
 ROOK	Rook	Rook configures Ceph using the operator model.

Understanding Scale - Approach



STEP 2:

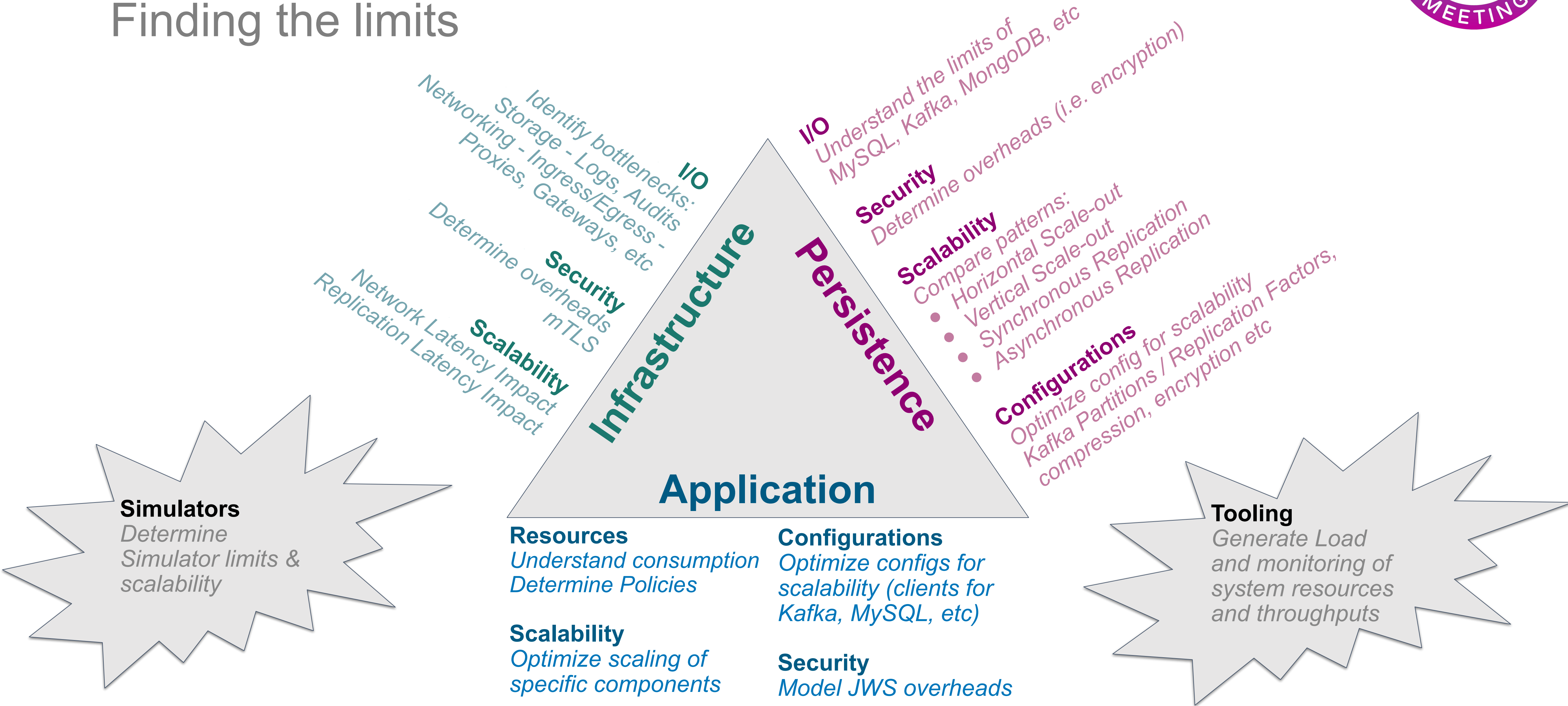
Define a pattern of *transactions* that constitutes a 'typical' workload

Principles:

- Transaction-traffic based on “busy” periods - PayDay, National Holiday etc.
- Generation of workload is to be outside the cluster-under-test.
- Limitations in specific software or hardware components may be accommodated for by scaling software horizontally.

Understanding Scale - Approach

Finding the limits



Understanding Scale - Approach



STEP 3:

Define a 'Standard' hardware pattern for non-critical deployment

Principles:

- Moderate specification aims for an achievable baseline.
- Customer-centric on-premises deployments.
- Vendor-agnostic, commodity hardware ensures availability to everyone.
- Failure-Protection in software - not hardware.

Understanding Scale - Approach

An example of an on-prem PoC



CPU	AMD Ryzen 5965WX 24 cores @ 4.5Ghz
RAM	4 x 32GB 3200MHz
DISK	1TB NVMe M.2 SSD
DISK	1TB NVMe M.2 SSD
NIC	Dual 10Gbs SFP+
NIC (option)	Dual 25Gbs SFP28
PSU	1000w

Unit Cost ~\$4000

Whilst high-end, this workstation-grade computer is available globally with lead-times < 8 weeks.

There is the inclusion for an optional 25Gbs network upgrade in cases where storage replication throughput becomes a bottleneck.

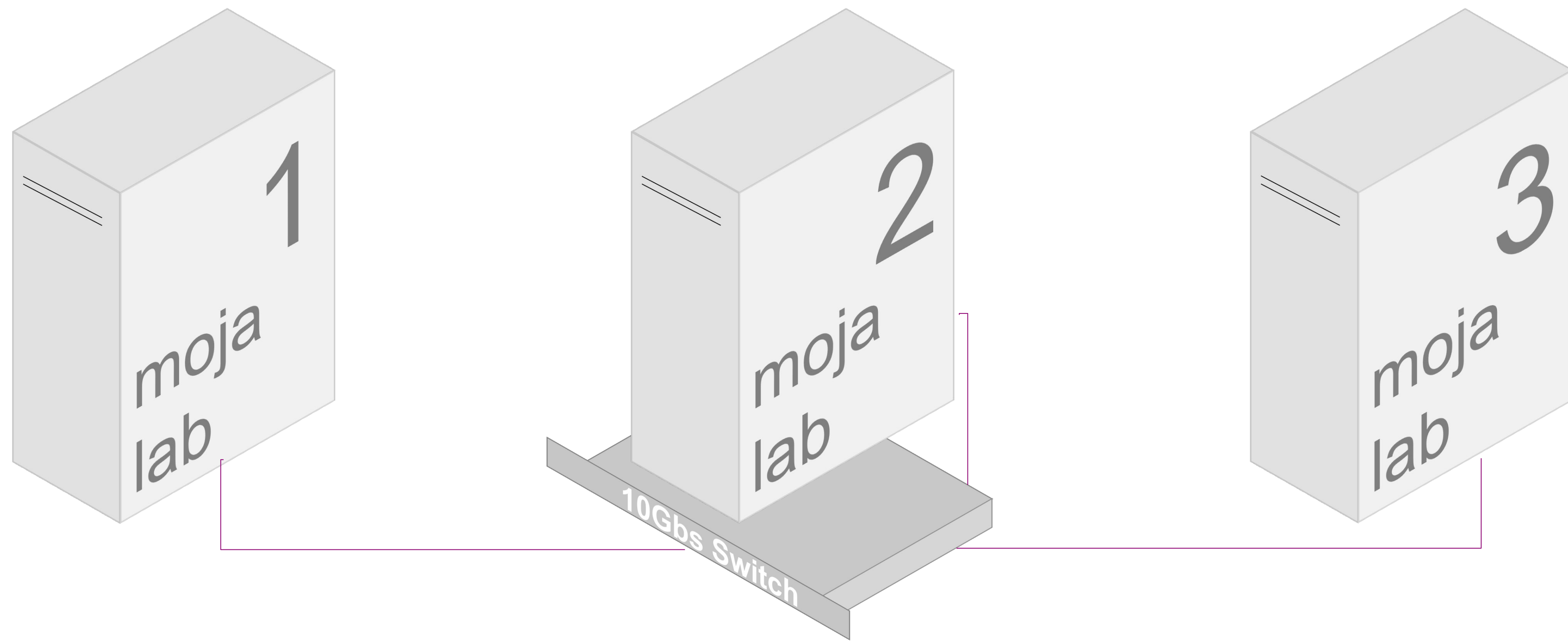
A pair of 1TB NVMe SSDs can be configured to run in RAID0 to maximise storage throughput, or as separate disks to mitigate pipelining.

The use of a Ryzen CPU ensures adequate PCIe bandwidth is provided to each device, whilst ensuring sufficient cores are available for both storage and compute duties.

Understanding Scale - Approach



For modelling customer workloads on-prem, three lab-scale nodes are connected with dual 10Gbs connections (optional 25Gbs) to the 10/25 Gbs switch. Storage replication connections are configured to a dedicated VLAN. Application traffic is configured to another. 1Gbs or 10Gbs uplinks from the switch can be connected to allow test-traffic to access the system in a friendly-user PoC. For performance testing, Node 3 can be configured to generate intense workloads against a replicating pair (1 & 2). Alternatively, a development team can run multiple, concurrent tests from their workstations.





Production Ready

Building a production solution

Bringing this to production



1. Standardized Testing

Today

2. Customer Configuration

3. Configuration Testing

4. Procure

Standardized Dataset and hardware testing in-cloud

- Develop mojaStack-IaC
- Understand which components need to scale
- See where bottlenecks are going to be.
- Deliver a standardized configuration to feed into customer-specific depoyments

Configure the mojaloop-IaC for your use-case

Inputs:

- How much throughput?
- Traffic shape
- Risk-Appetite

Result:

A baseline configuration specific to customers needs

Conduct a friendly-user PoC against the proposed configuration in-cloud and on-prem hardware

- Adjust configuration to optimize customer specific workload.
- Verify scaling & availability requirements can be met.

Results inform purchasing BoM

Test results are used to build:

- Infrastructure Scaling Plan
- Phased Bill-of-Materials
- Scale-out Business Case

Q & A

- Thank You for listening



Understanding Availability

Is N+1 enough: Active + Active

Under Normal Operation

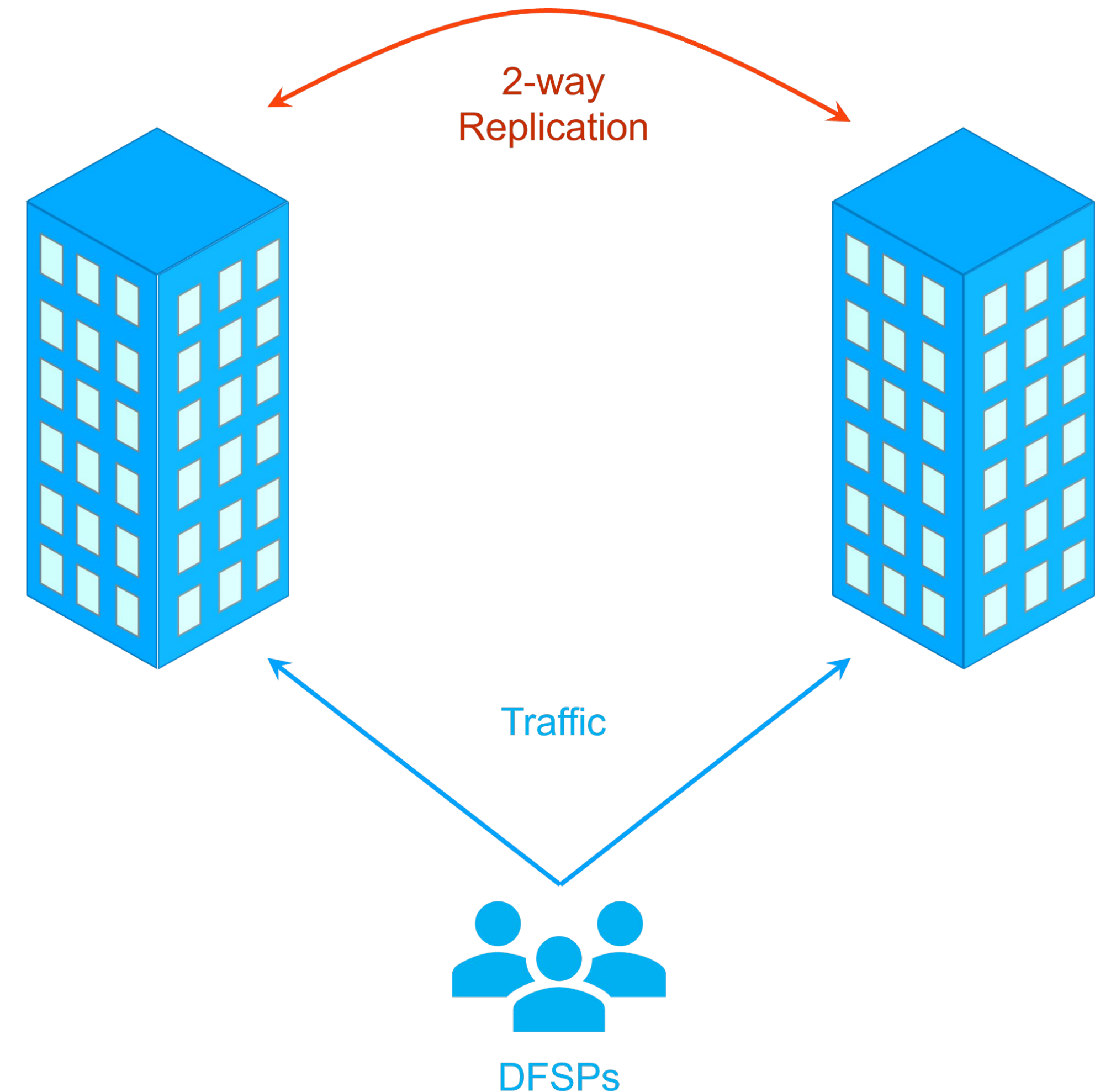
- Every transaction must wait for replication (issues in one DC may still affect the other).
- Maintenance requires pausing replication **and** incoming traffic to target DC.
- 50% of each DC must be reserved for peak transaction flow.
- Failover must be tested often.

During a controlled failover

- No interruption to transaction traffic – DC ‘drains’ inbound connections.
- In-Flight transactions will finish.
- Best performed during periods of low-throughput.

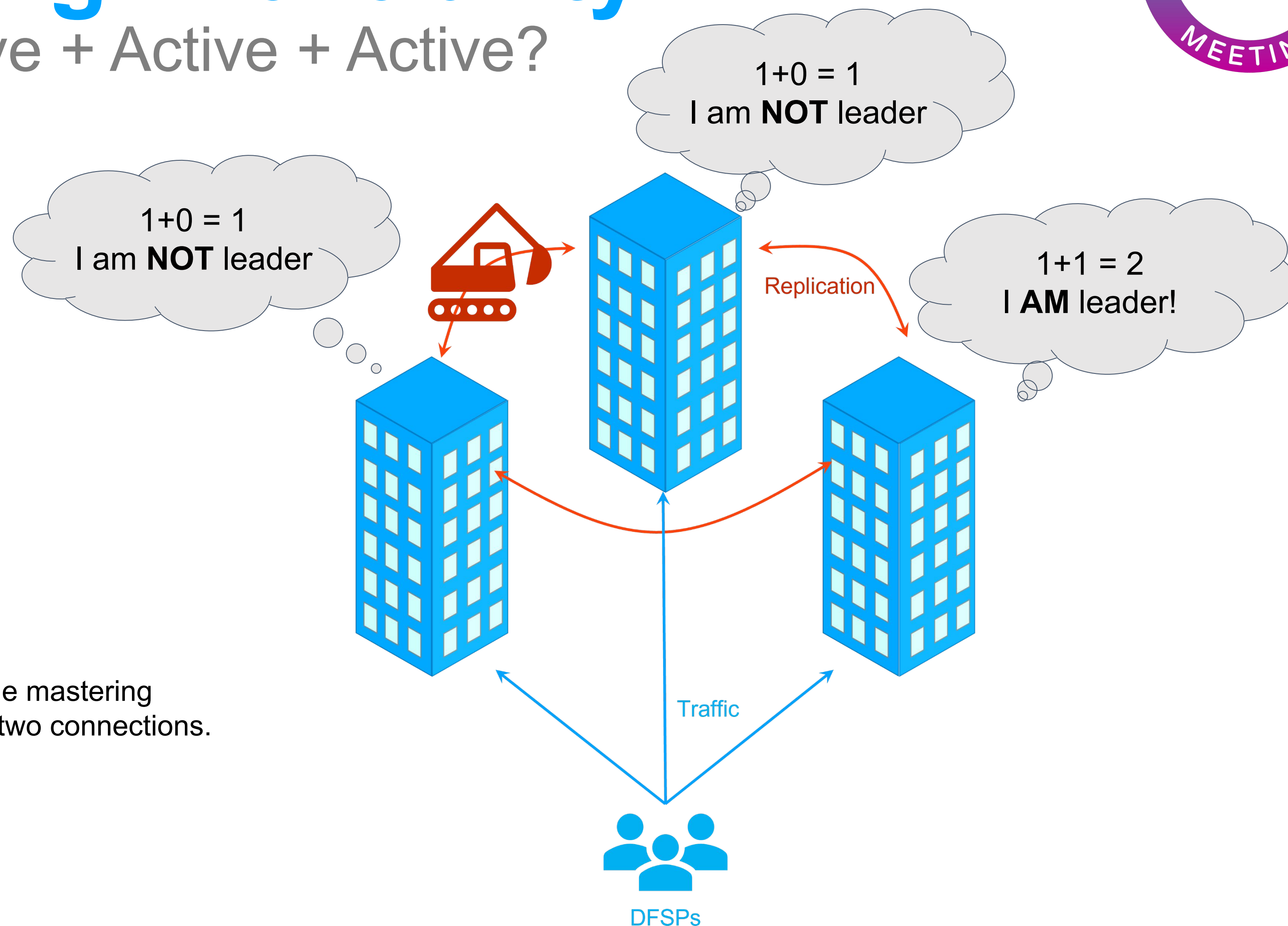
During an uncontrolled failover

- In-Flight transactions to failed DC will be lost.
- In-Flight transactions to healthy DC will pause.
- Likely to require **Manual Failover**.
- **Significant Risk of ‘Active-Active Split-Brain’**.



Understanding Availability

Is N+1 enough: Active + Active + Active?



During a network partition:

- All 3DCs can see 1 other DC.
 - 1DC can see 2DCs.
- Replication between all 3 DCs continues.
- Any DC with one connection will respect the mastering decision of its partner DC, which will have two connections.
- Performance need not be impacted.

Understanding Scale - Methodology



Characterization of the System and its Scalability will need to be addressed by the following pillars:

- **Infrastructure** ← *Understand the underlying Infrastructure limits*
 - **I/O** ← *Understand the general Input-Output limits of Storage (i.e. Logs), and Networking (i.e. Ingress/Egress - Proxies, Gateways, etc) limits in isolation*
 - **Security** ← *Determine overheads (i.e. TLS)*
 - **Scalability** ← *Understand the impact of scaling Infrastructure (i.e. Network latencies)*
 - what other factors can we add here?
- **Persistence Stores** ← *Understand the Persistent limits*
 - **I/O** ← *Understand the limits of our Persistent Stores (i.e. MySQL, Kafka, MongoDB, etc) in isolation*
 - **Scalability** ← *Determine the best scaling approach (i.e. master-master vs master-slave, horizontal vs vertical, etc) and their impact*
 - **Configurations** ← *Determine the required configurations to support scalability (i.e. Kafka Partitions / Replications Factors, compression, encryption etc)*
 - **Security** ← *Determine overheads (i.e. encryption)*
 - what other factors can we add here?
- **Application** ← *This is needed to ensure that we can scale predictably and in a stable fashion*
 - **Resources** ← *Understanding resource usage, and determining how to set Resource Policies.*
 - **Scalability** ← *Determine which parts of the system are scalable, and their scalability approach (i.e. horizontal vs vertical, scaling policies)*
 - **Configurations** ← *Determine Application configurations to support the scalability (i.e. MySQL, Kafka client configurations - message compression, etc)*
 - **Security** ← *Determine overheads (i.e. JWS)*
 - **Simulators** ← *Determine Simulator limits & scalability*
 - what other factors can we add here?