dCache.org

dCache in a Cloud

Tigran Mkrtchyan

Nordic e-Infrastructure Collaboration

eXtreme DataCloud

LSDMA

Fermilab

DESY.

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

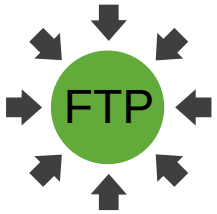
Three problems to solve

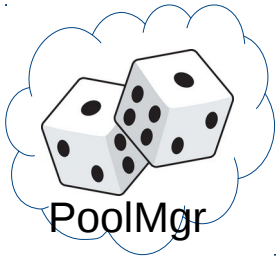
- Data never fits into a single server
  - Multiple servers
  - Off-load to tape
- Growing number of clients
  - Main frame vs. Linux cluster
- We want our own HW/OS selection
  - Better offers
  - Local expertise

Four main components

- **DOOR**
  - user entry points (NFS, FTP, DCAP, XROOT)
- **POOL**
  - data storage nodes, talk all protocols
- **Namespace**
  - metadata DB, POSIX layer
- **PoolManager**
  - request distribution unit

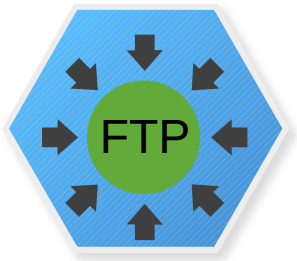
Minimal Setup



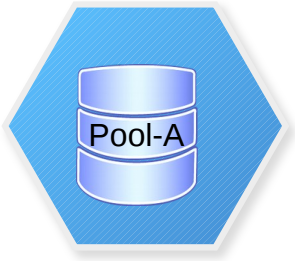


Minimal Setup

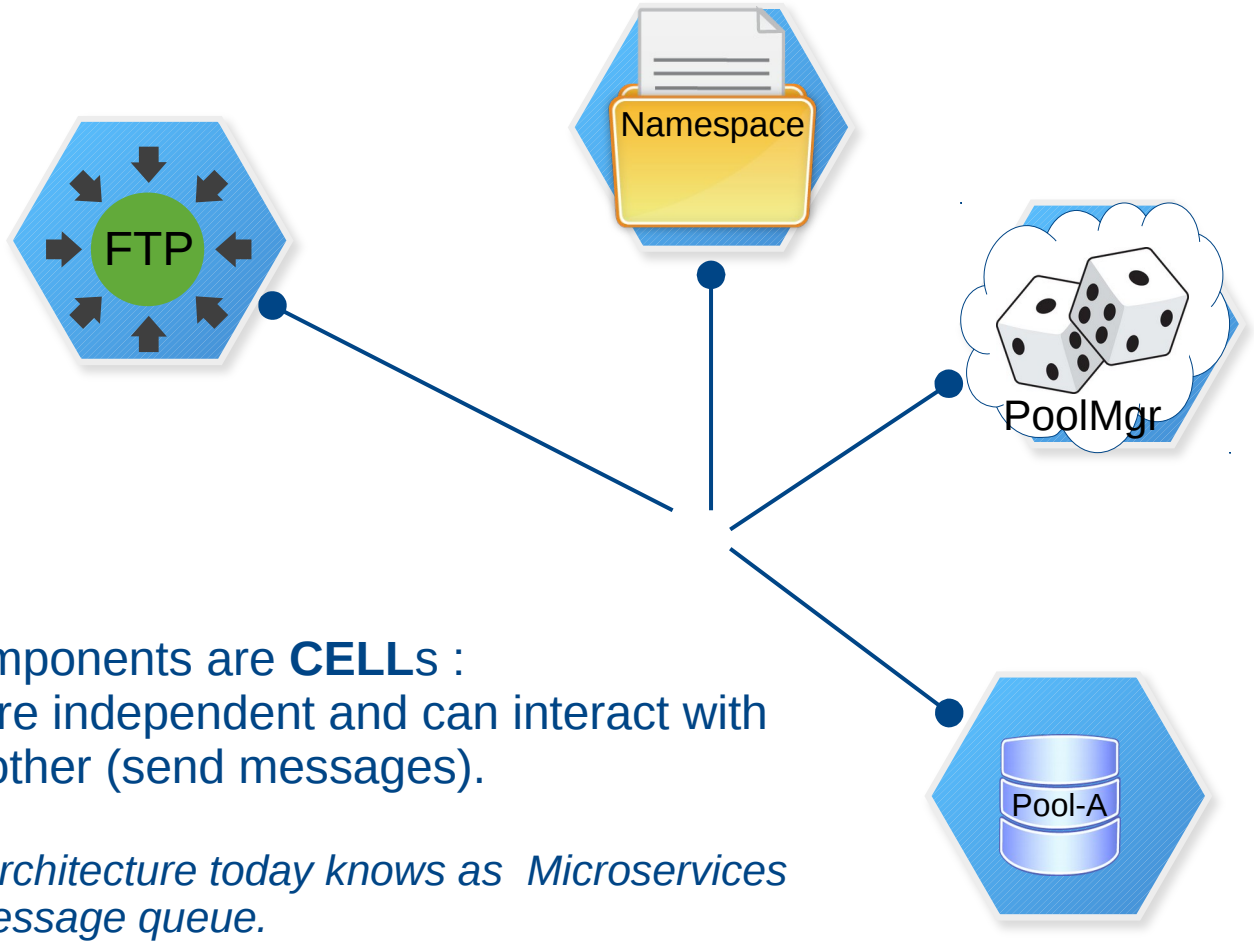
Minimal Setup



All components are **CELL**s :
they are independent and can interact with each other (send messages).



Minimal Setup

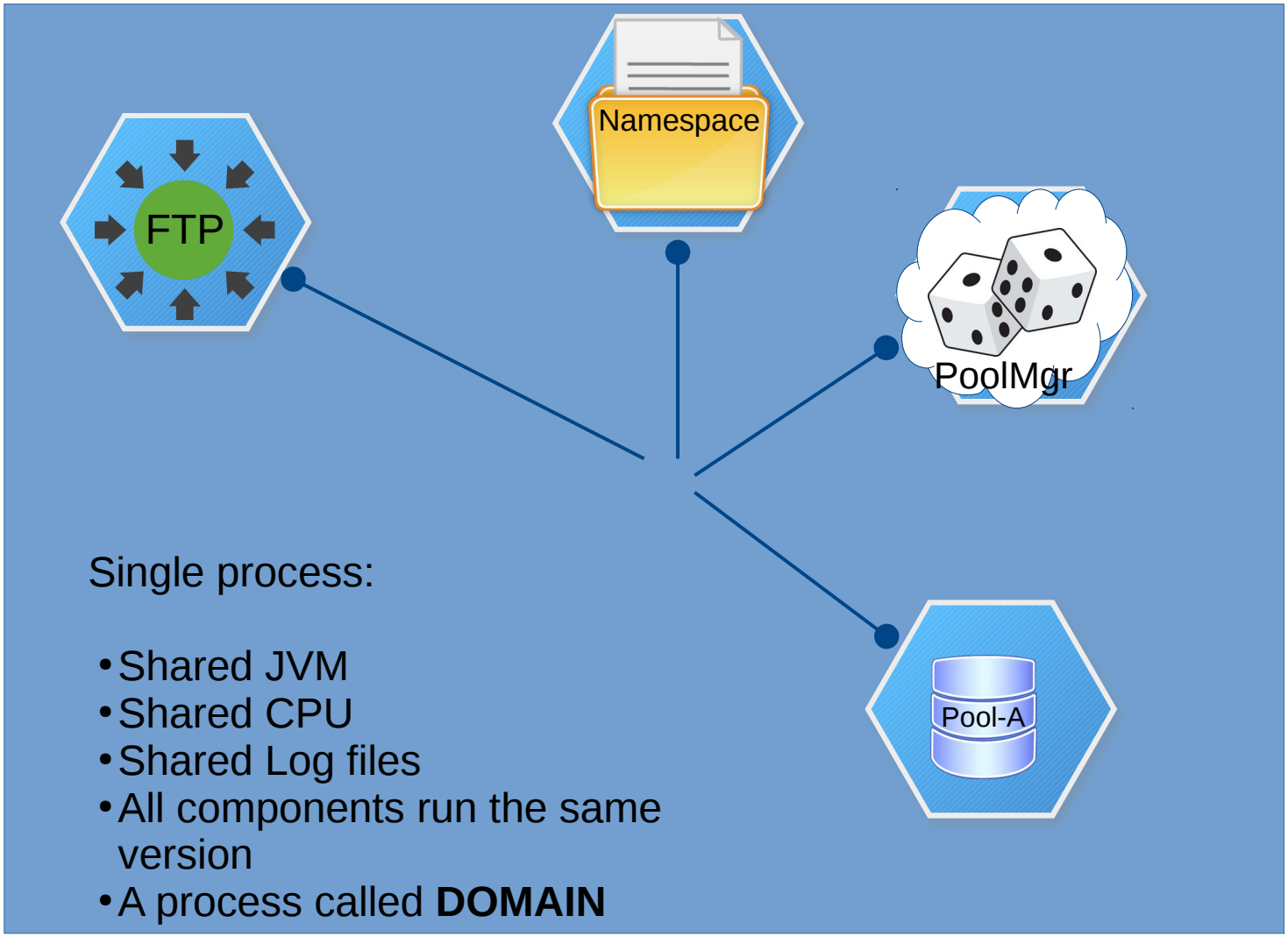


All components are **CELL**s :
they are independent and can interact with each other (send messages).

Such architecture today knows as Microservices with message queue.

Grouping CELLs

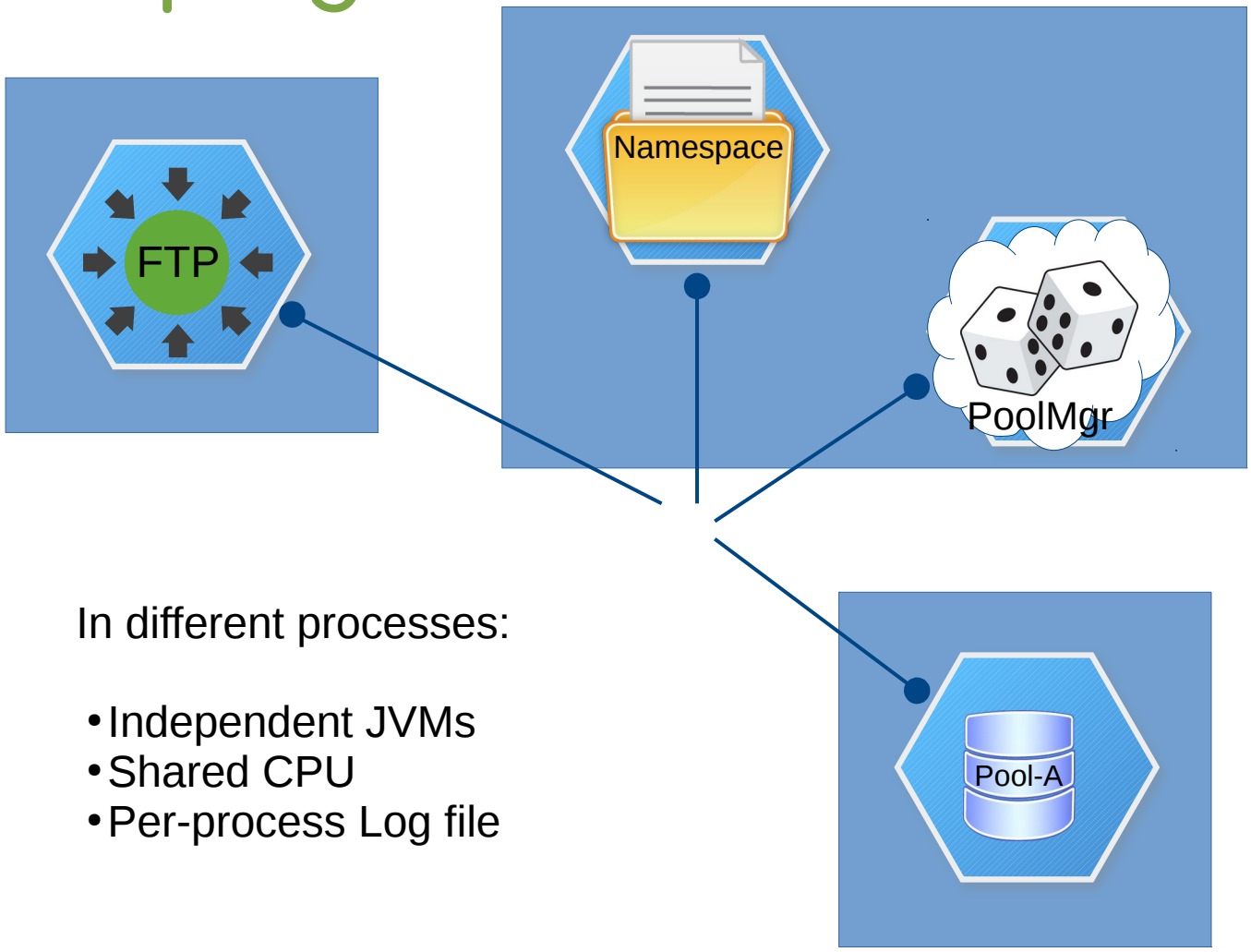
Grouping CELLs



Single process:

- Shared JVM
- Shared CPU
- Shared Log files
- All components run the same version
- A process called **DOMAIN**

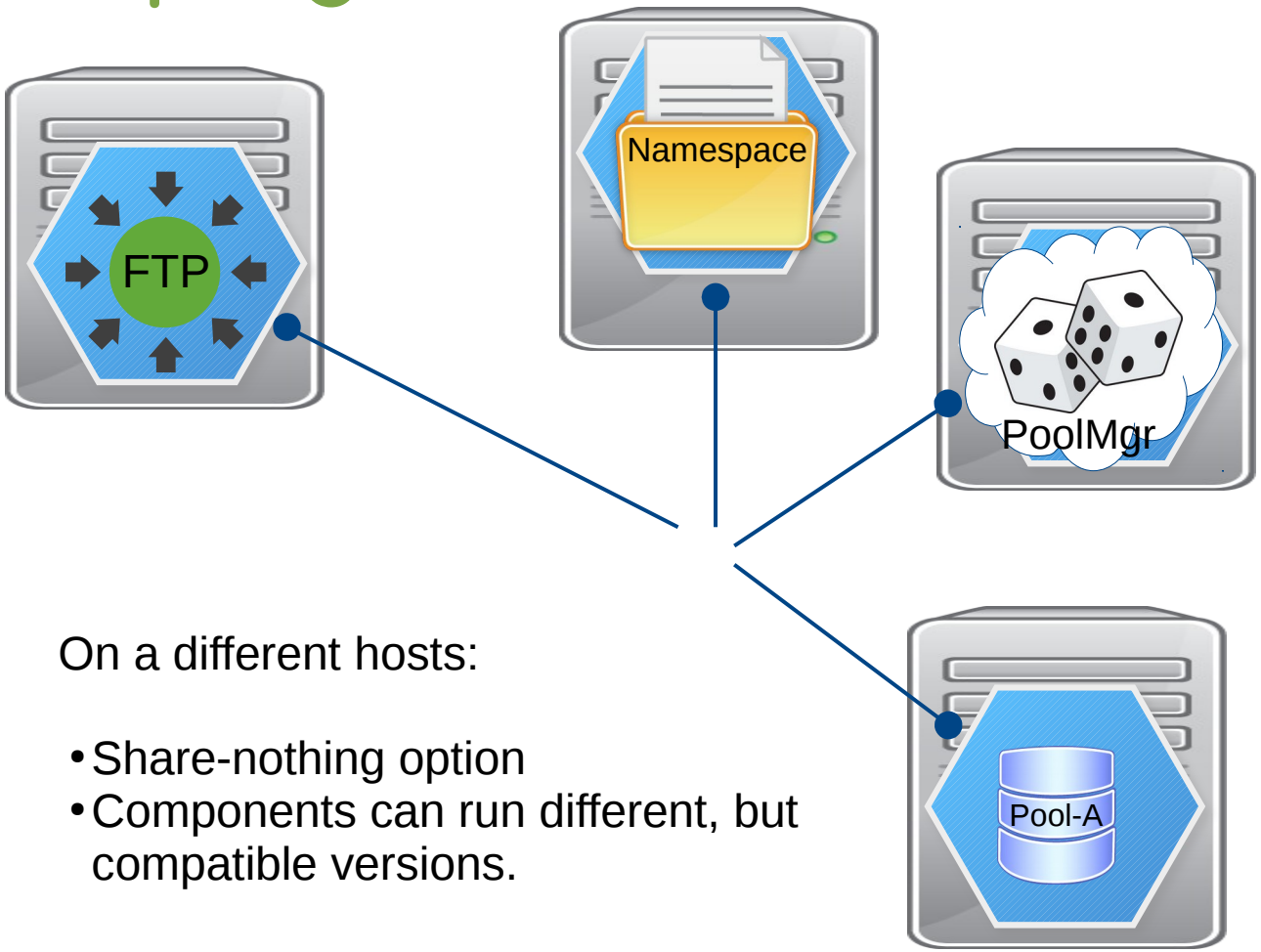
Grouping CELLs



In different processes:

- Independent JVMs
- Shared CPU
- Per-process Log file

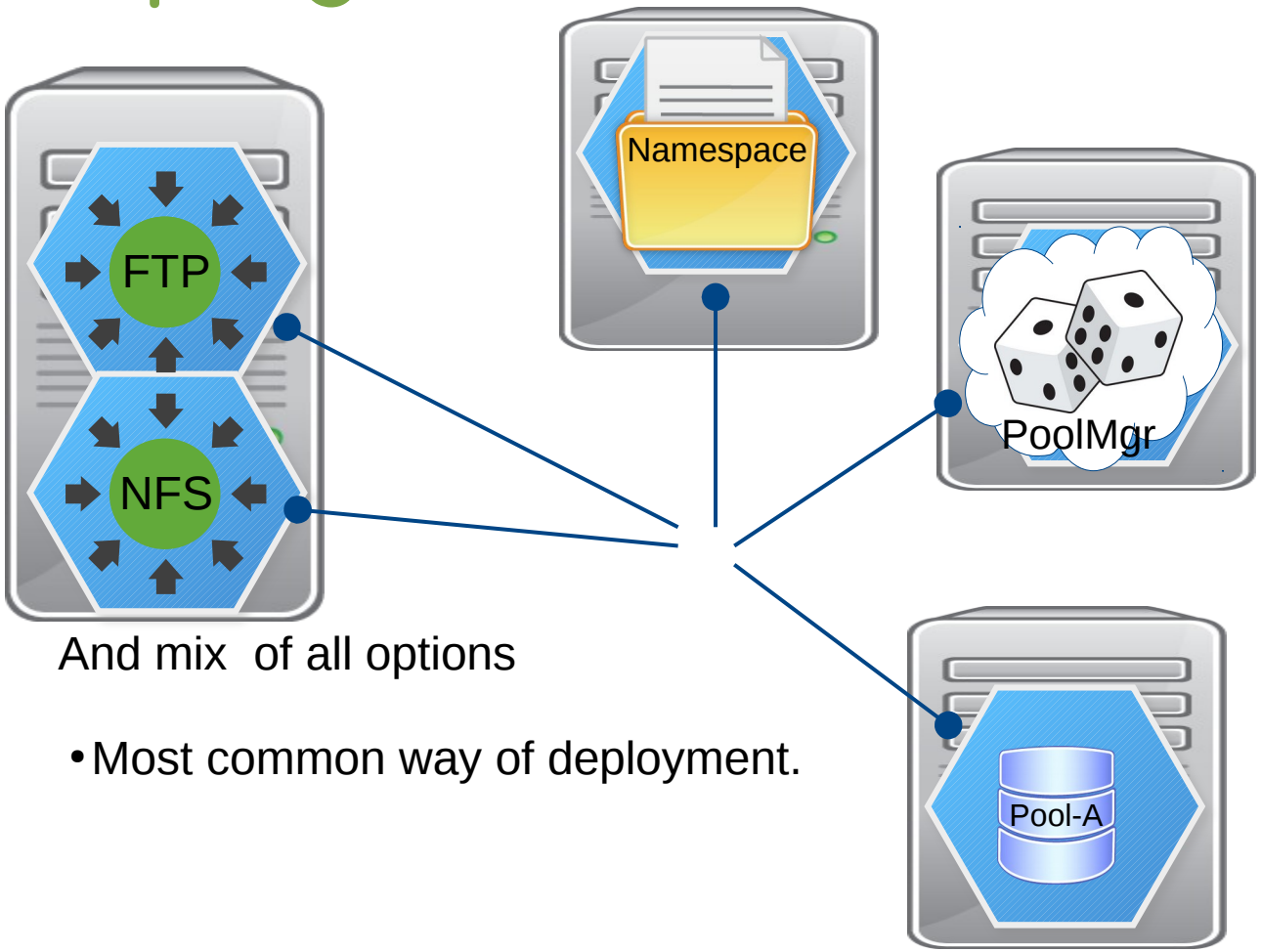
Grouping CELLs



On a different hosts:

- Share-nothing option
- Components can run different, but compatible versions.

Grouping CELLs



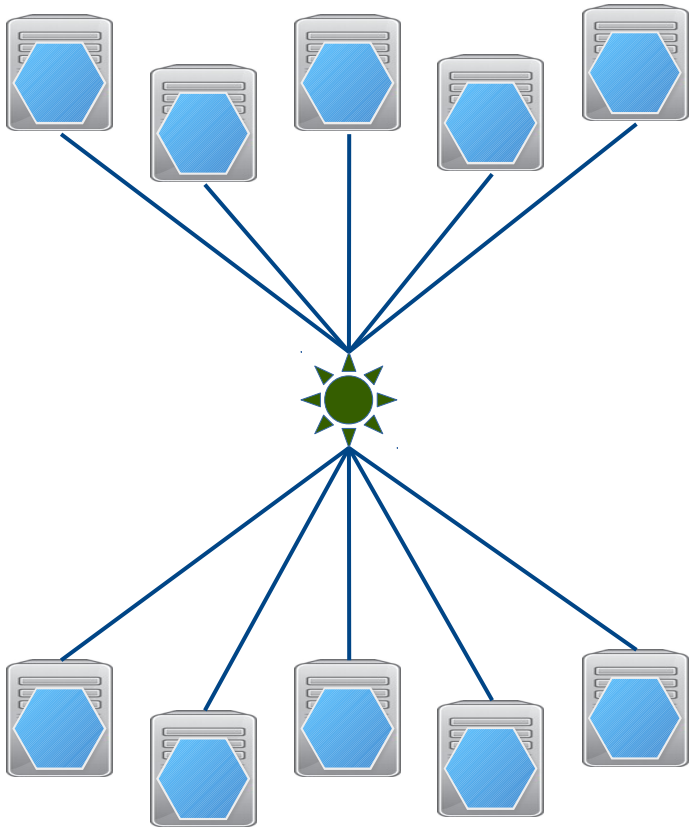
And mix of all options

- Most common way of deployment.

Internal Communication

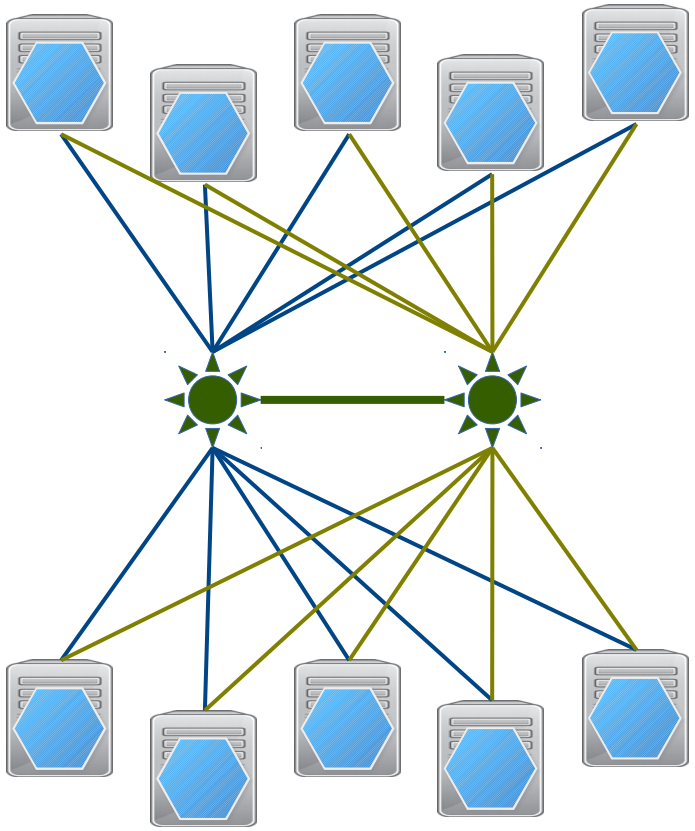
- Inter-cell communication
  - Message passing
  - Routing
- ZooKeeper
  - Service discovery
  - Coordination
  - Configuration

Cell messaging 101



- Star like topology
- Selected node configured as a hub called **CORE** domain
- All communication goes through CORE domains
- Other domains called **SATELLITE**

Cell messaging 101

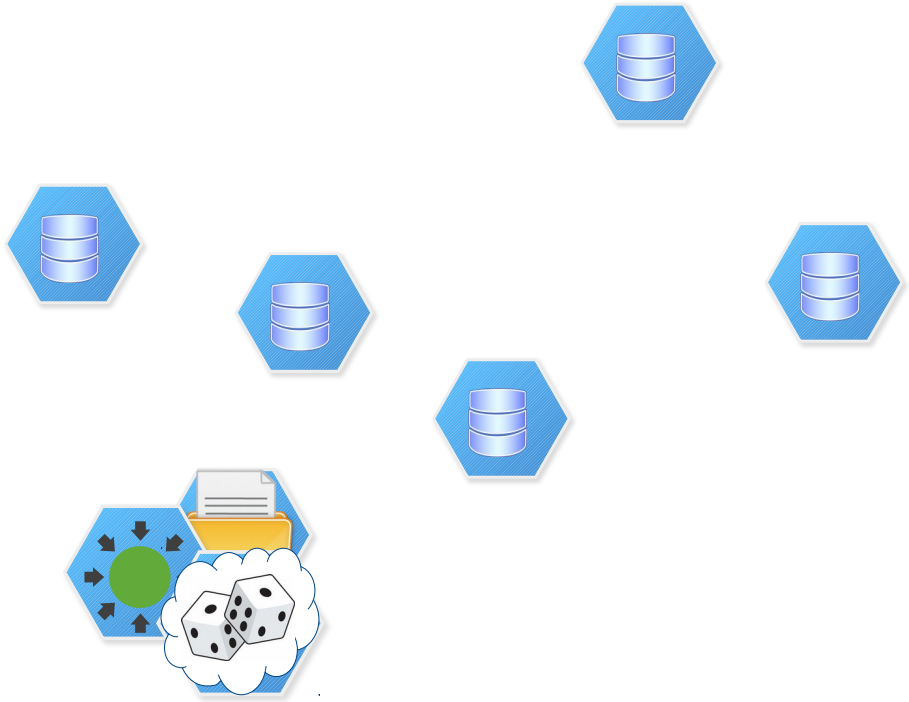


- Star like topology
- Selected node configured as a hub called **CORE** domain
- All communication goes through **CORE** domains
- Multiple **CORE** domains make communication fault tolerant

Multi-Site deployment

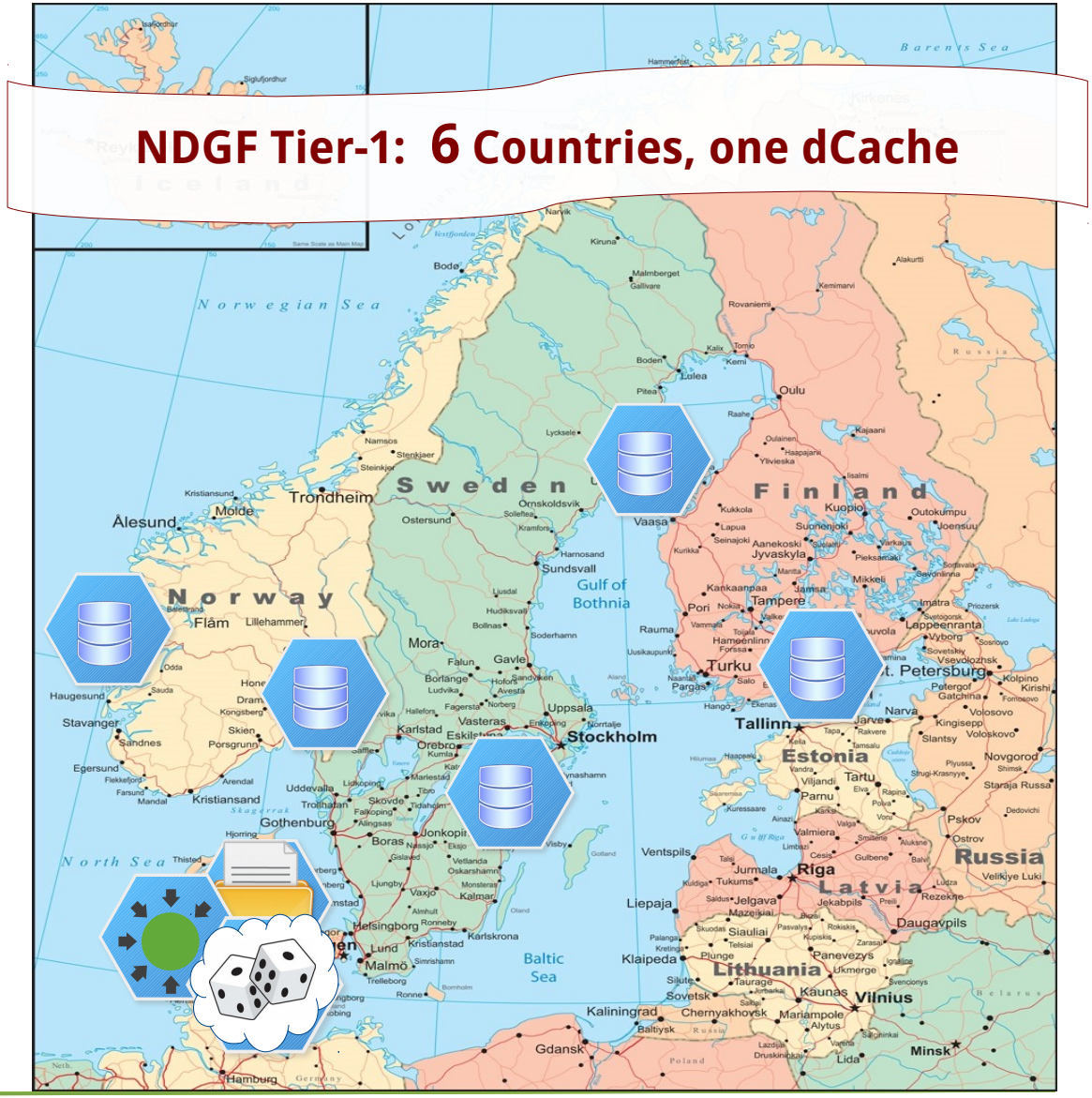
Multi-Site deployment

- Distribute data over multiple locations
- Multiple administrative domains
- Use available resources



Multi-Site deployment

- Distribute data over multiple locations
- Multiple administrative domains
- Use available resources



Multi-Site deployment

- Works for all protocols
- Support HSM connectivity
  - Each site/pool may have it's own tape system
- Pools may run different major versions
  - Site has two years to upgrade pools

Multi-Site deployment

- Preferred write location depending on IP (location) or directory path (if requested)
- Preferred 'local' read access if data is available
- Replication
  - On Demand, when requested from remote site
  - Permanent, data protection, location adjustment
  - Manual, for data location optimization, maintenance

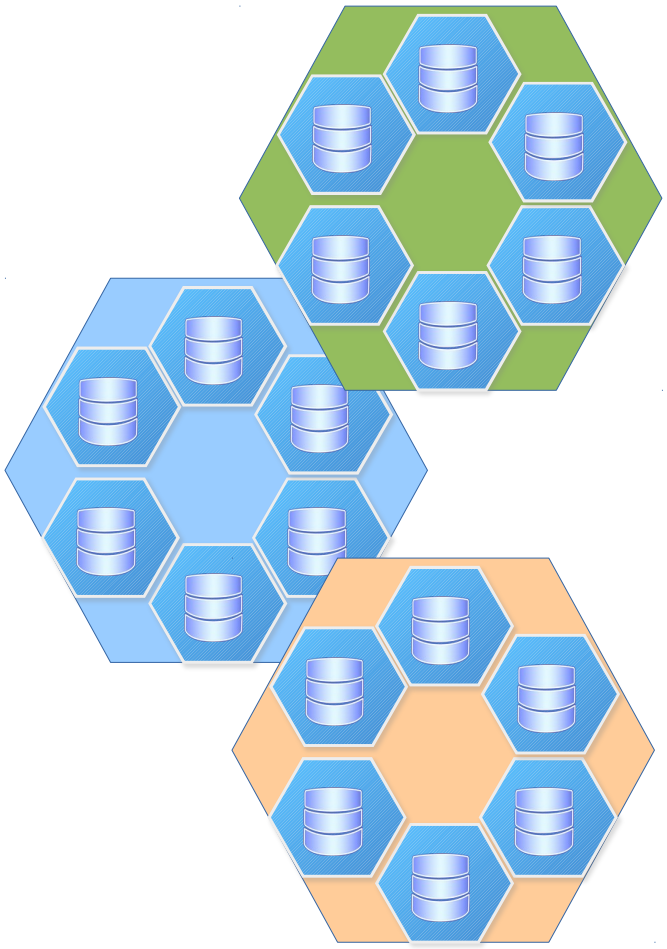
Multi-Site deployment

Pool selection internals

Request Parameters

- Protocol
- Path
- Client IP

Pool Groups

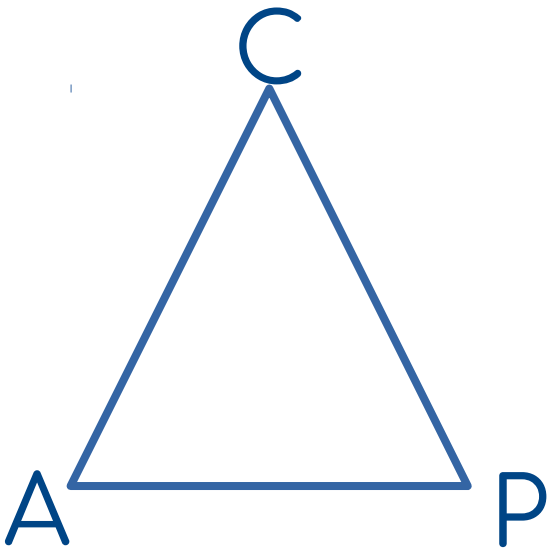


CAP

- Consistency
  - Every read receives the most recent write or an error.
- Availability
  - Every request receives a (non-error) response – without guarantee that it contains the most recent write.
- Partition tolerance
  - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

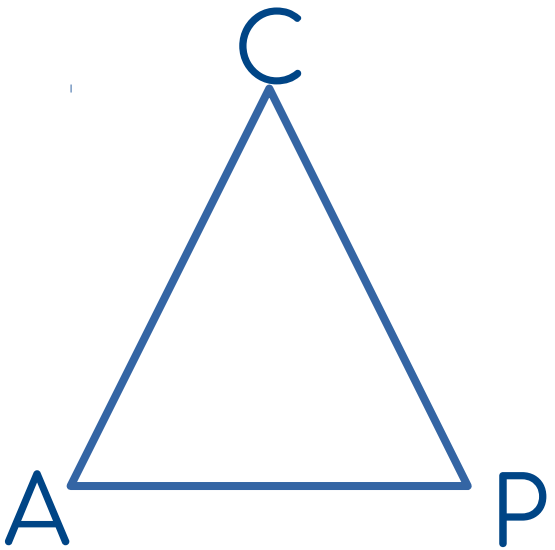
The CAP Theorem:

You can have at most two of these properties for any shared-data system



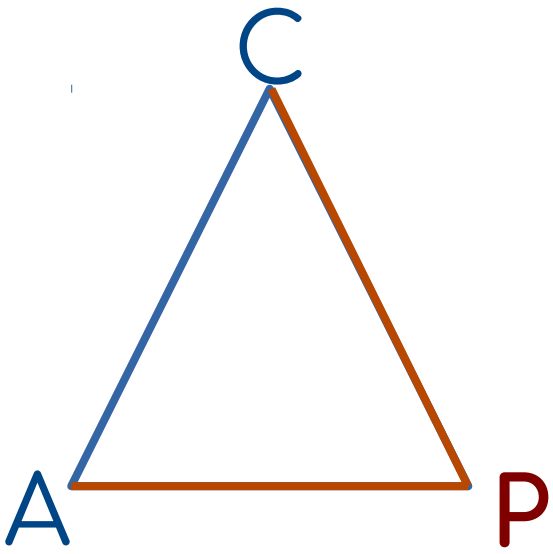
The CAP Theorem:

~~You can have at most two of these properties for any shared-data system~~



CAP theorem explained

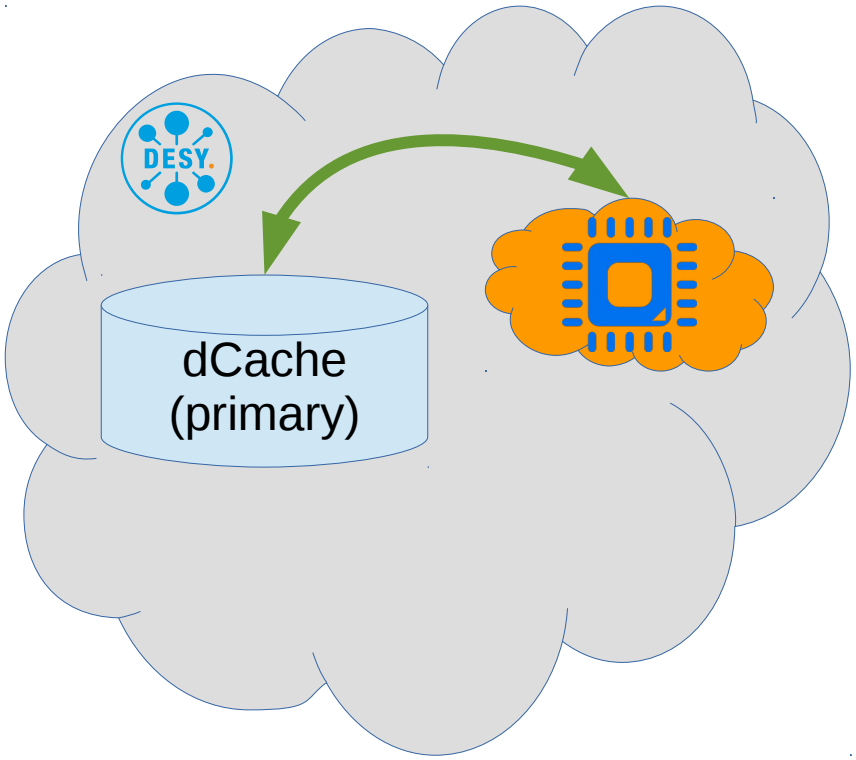
- No distributed system is safe from network failures.
- In the absence of network failure both availability and consistency can be satisfied.
- The choice is between consistency and availability only when a network partition or failure happens.



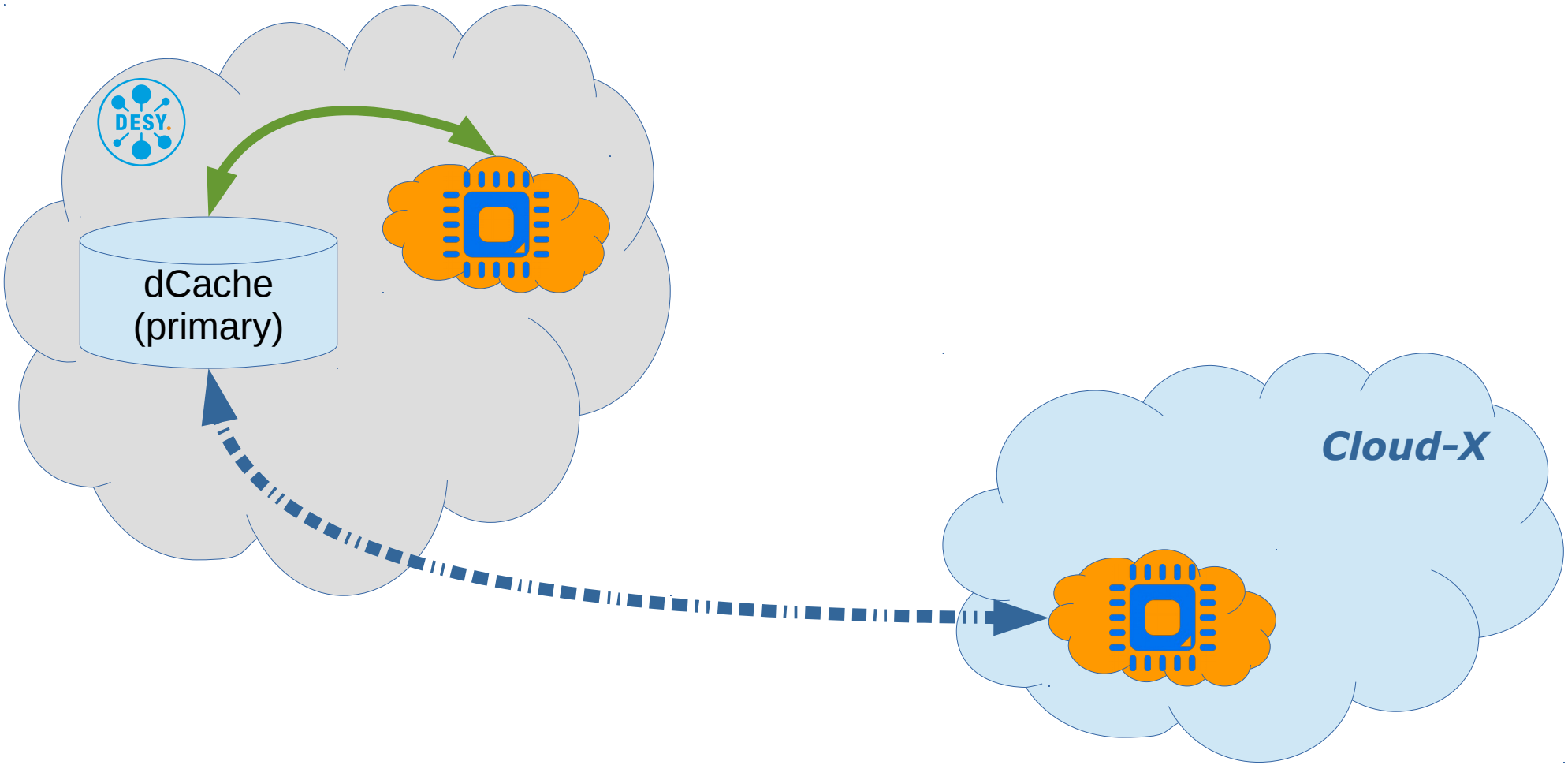
dCache and CAP

- dCache provides consistency over availability.
- All client will see the same data at the same time.
- A timeout or error will be returned, if consistency can't be guaranteed.

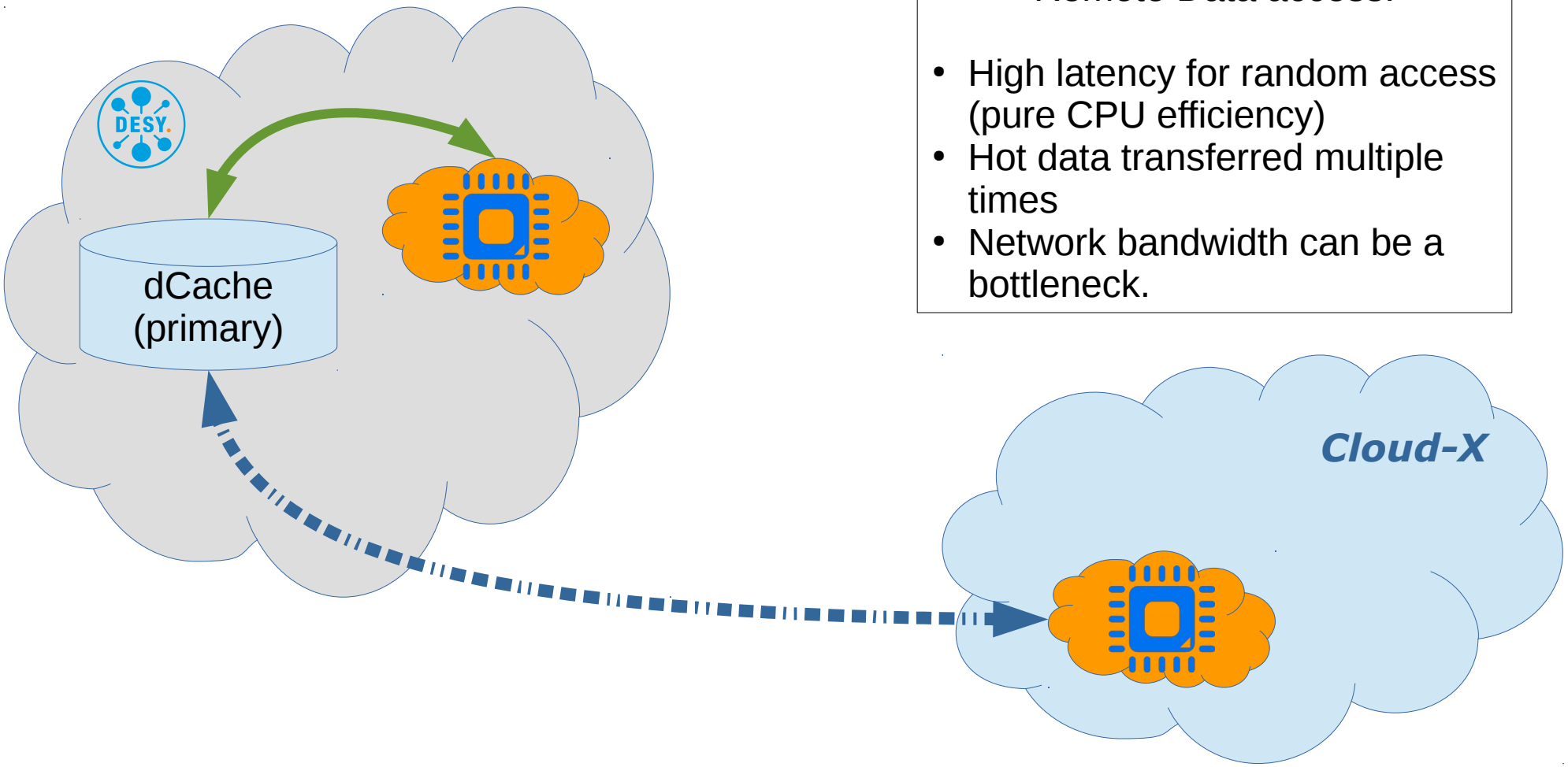
Cloud deployment (scenario I)



Icon made by Freepik from www.flaticon.com



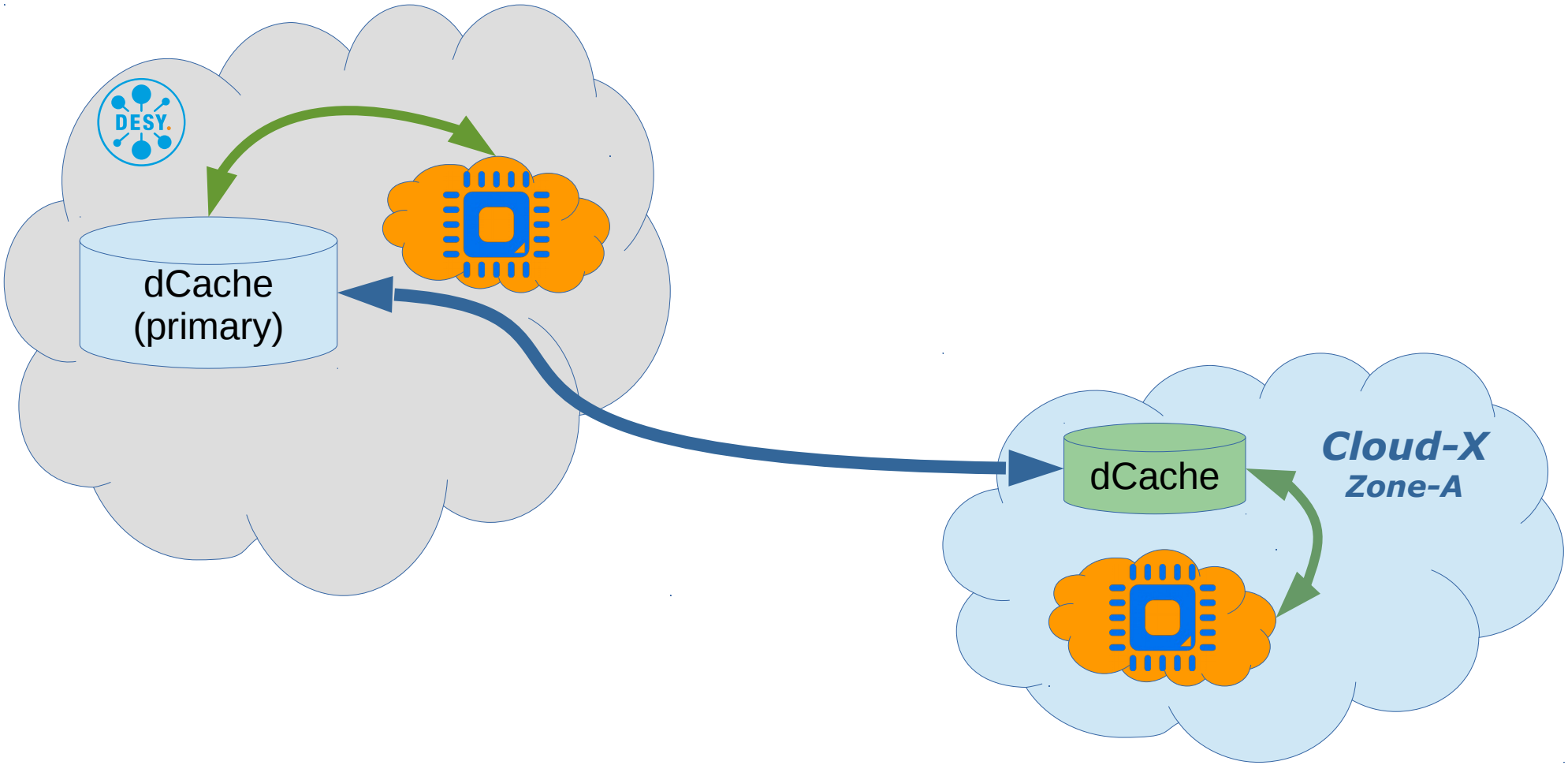
Icon made by Freepik from www.flaticon.com



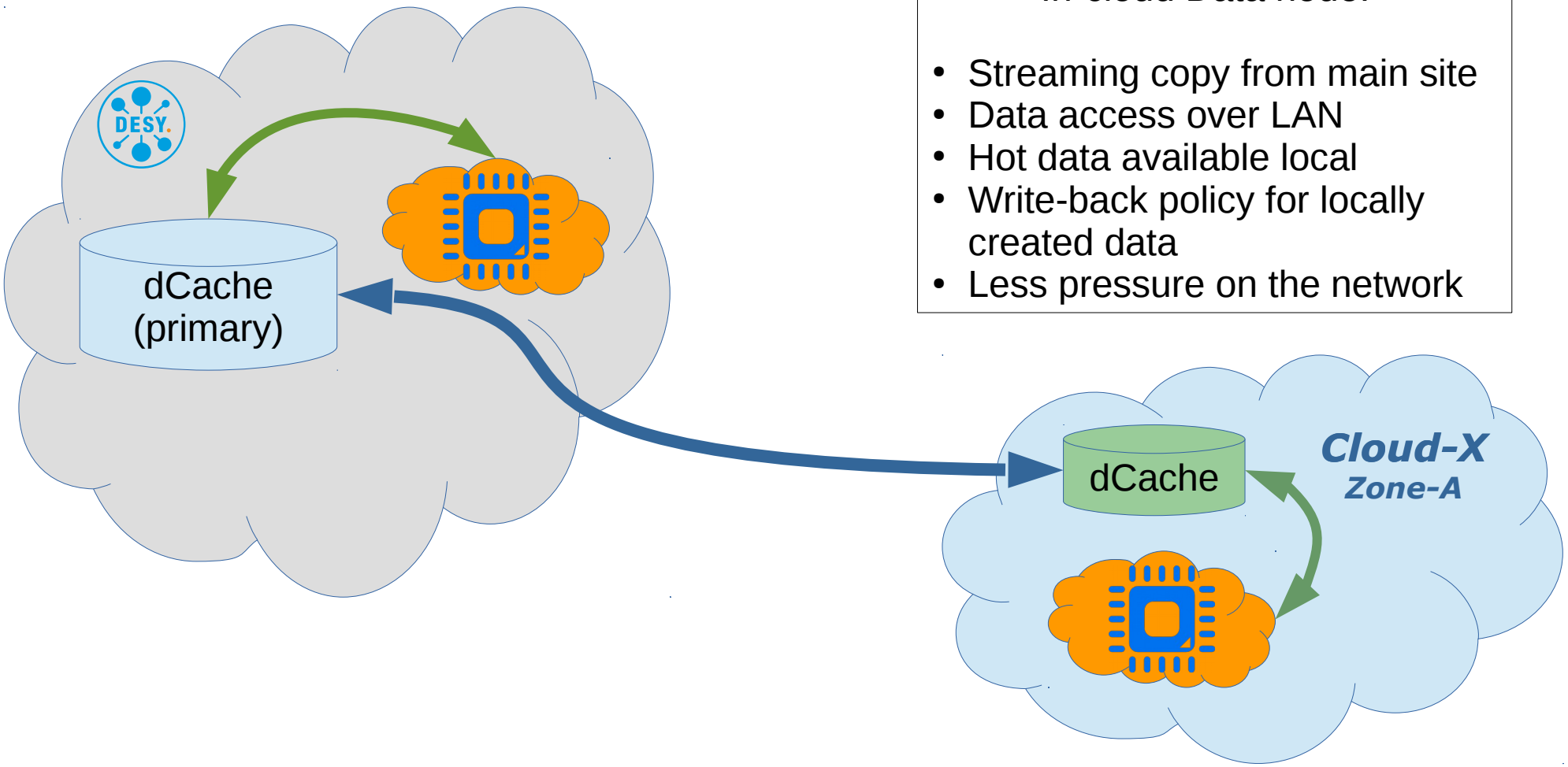
Remote Data access:

- High latency for random access (pure CPU efficiency)
- Hot data transferred multiple times
- Network bandwidth can be a bottleneck.

Icon made by Freepik from www.flaticon.com



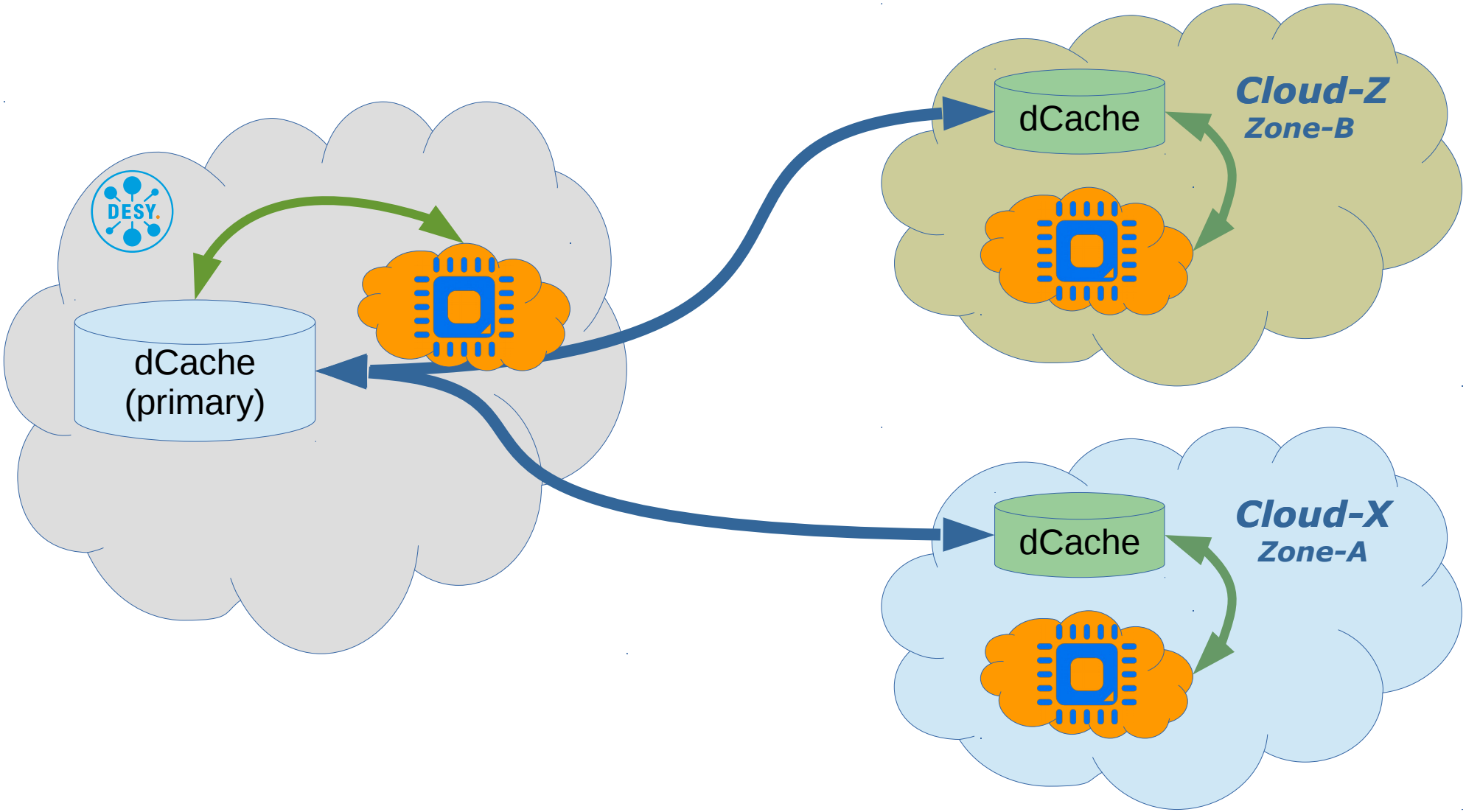
Icon made by Freepik from www.flaticon.com



In-cloud Data node:

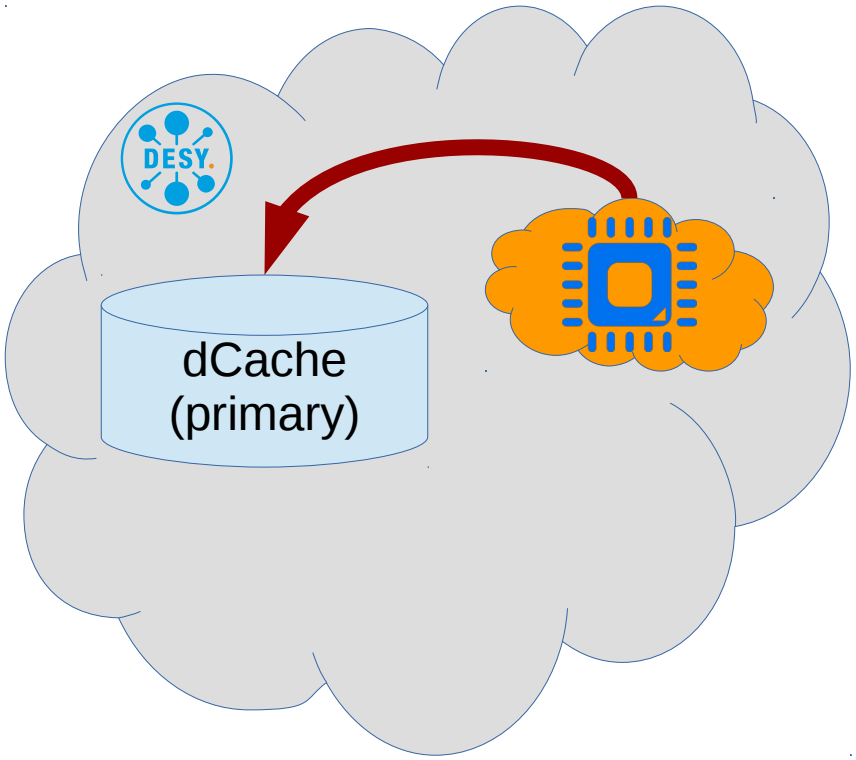
- Streaming copy from main site
- Data access over LAN
- Hot data available local
- Write-back policy for locally created data
- Less pressure on the network

Icon made by Freepik from www.flaticon.com

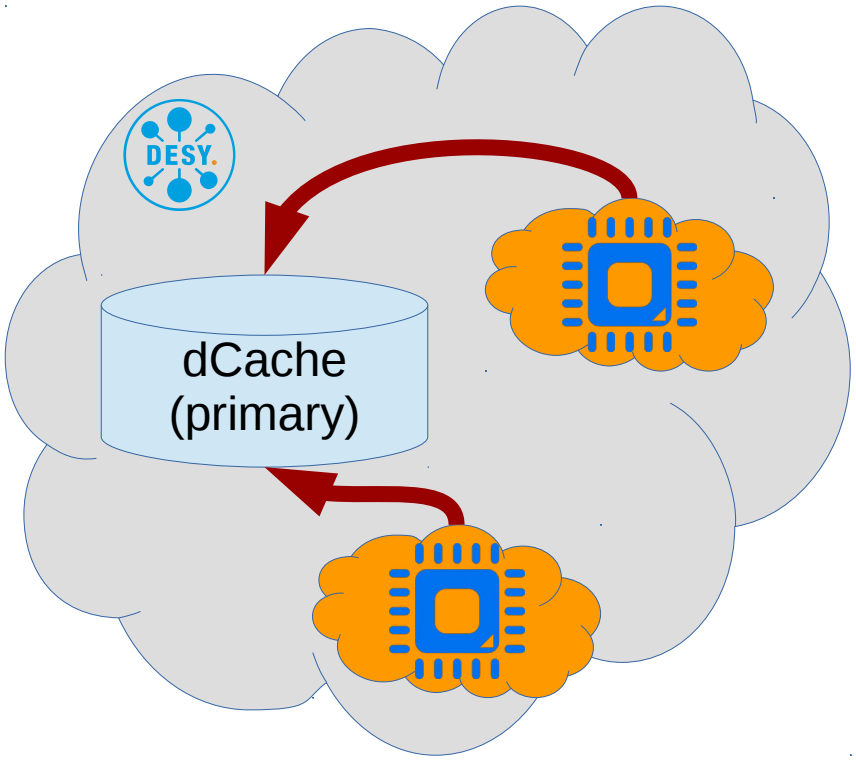


Icon made by Freepik from www.flaticon.com

Cloud deployment (scenario II)

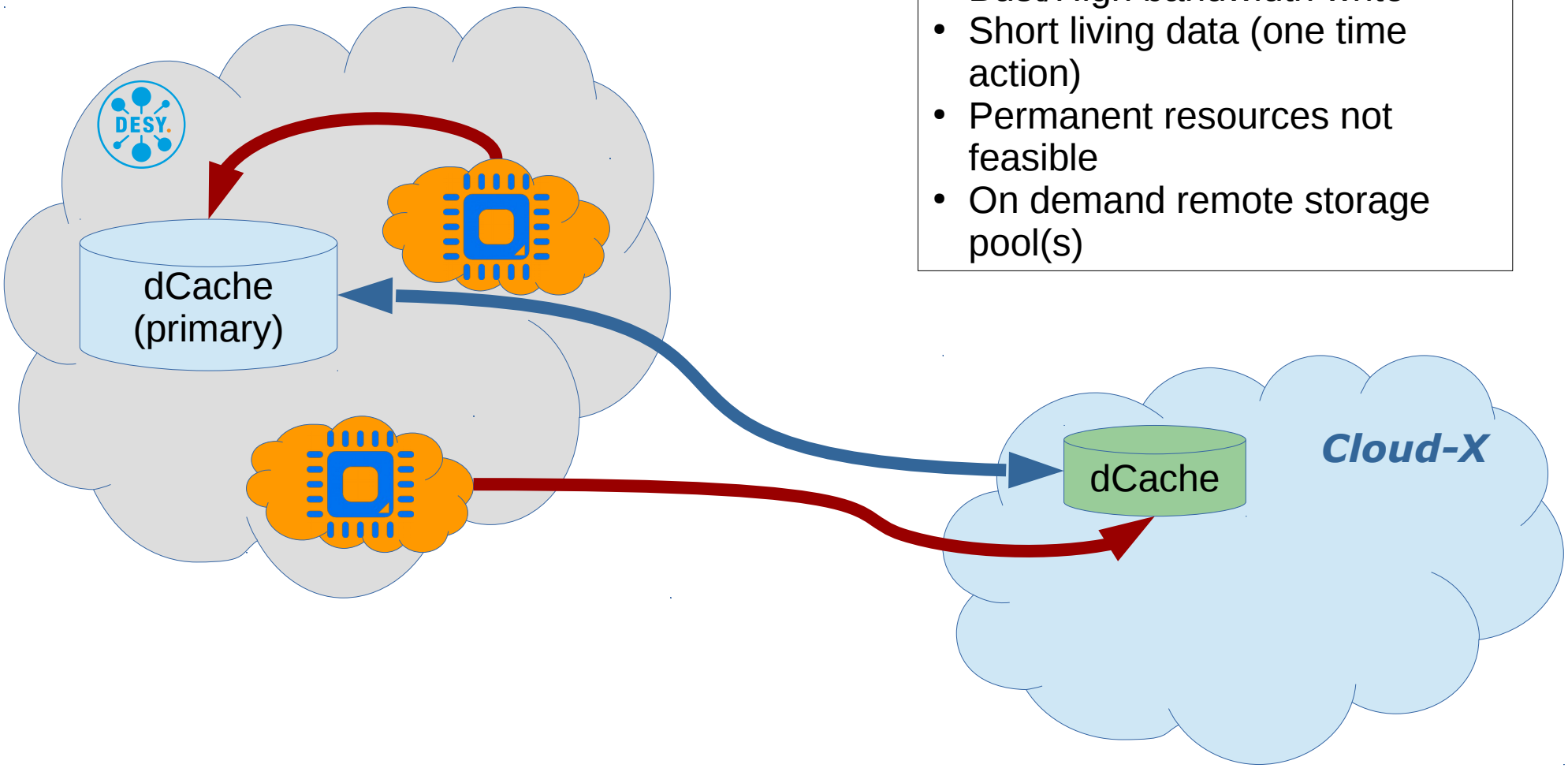


Icon made by Freepik from www.flaticon.com



- Bust/High bandwidth write
- Short living data (one time action)
- Permanent resources not feasible

Icon made by Freepik from www.flaticon.com



- Bust/High bandwidth write
- Short living data (one time action)
- Permanent resources not feasible
- On demand remote storage pool(s)

Icon made by Freepik from www.flaticon.com

Requirements

- IP based separation of different Clouds or Zones required.
- Should be possible to start data server in desired zone.
  - API
- Data internal communication protection requires VPN or host certificates on demand.