Reference Architecture for SAP HANA

Objective

SAP HANA database is offering many different options how to design the infrastructure.

There are different ways how to implement High Availability (HA) and Disaster Recovery (DR) and there are many optional SAP HANA extensions (like Extension Nodes, Dynamic Tiering, XSA, etc.) that can be deployed.

There are various considerations that must be taken into account when designing infrastructure - for example ability to seamlesly move tenant (tenant portability) or whole instance (instance portability) without breaking external connectivity to the component.

Additional challenge is how to configure hostname resolution for individual virtual IPs to enable support for certificates and ensure their validity in relation to tenant or instance portability.

As result of this there are too many ways how to deploy SAP HANA and therefore almost all published Reference Architectures are very high-level showing only the generic concepts and they lack important details how to correctly implement the solution.

Goal of this project is to provide **one standardized multi-cloud and on-premise architecture** for SAP HANA that is able to support as many optional extensions as possible and to offer implementation details for different platforms including Public Cloud vendors (AWS, Azure, GCP, etc.) as well as on-premise implementations (VMware, Bare Metal).

It is important to state that other architectures are still valid (as long as formally supported by SAP) and can be used for specific requirements or use cases.

Approach

The approach taken by the team is driven by the opinion that it is more simple to remove the features rather than to add them and make them working in harmony with the rest of the design.

Basic steps are following:

1. Define complex requirements including most common optional features
2. Make Architectural Decisions (ADs) to reduce the amount of deployment options
3. Design infrastructure architecture meeting as many requirements as possible (one standardized architecture)
4. Derive simplified versions of the architecture by removing specific requirements
5. Modify the infrastructure architecture for different platforms by introducing platform-specific details

Table of Content

Contributing

Please refer to How to Contribute to understand how to contribute to this project.

Change Log

2019-08-08

- [Tomas Krojzl] Written initial content of `CONTRIBUTING.md` file
- [Tomas Krojzl] Written initial content of `README.md` file

2019-08-07

- [Tomas Krojzl] Repository created

How to Contribute

If you want to contribute to a project and make it better, your help is very welcome. Please see below instructions how to contribute.

1. Initial Setup

Perform following section only once at the beginning of your contribution.

1.1. Create GitHub user

Detailed instructions are here: https://help.github.com/en/articles/signing-up-for-a-new-github-account

1.2. Add SSH key

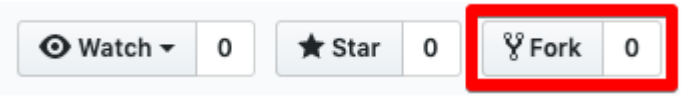
Adding SSH key will enable password-less connectivity to GitHub.

Detailed instructions are here: https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account

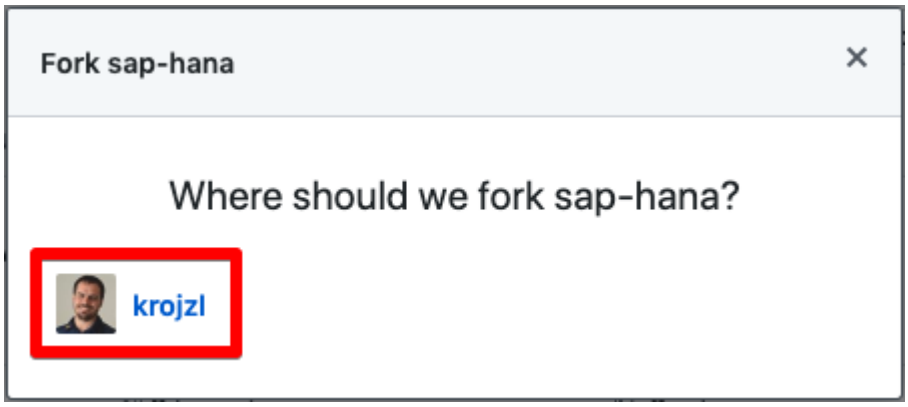
1.3. Fork the repository

Forking the repository will create your own personal copy of the repository.

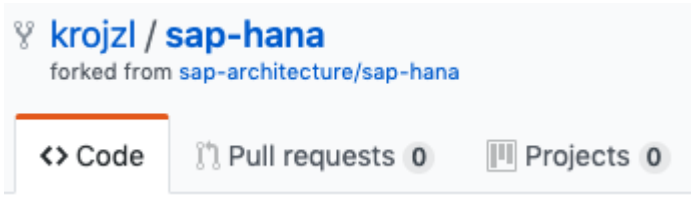
1. Navigate to the project repository: https://github.com/sap-architecture/sap-hana
2. Click on `Fork` button in upper-right corner of the page



3. Select your user to create copy in your private space



4. As result you should see that repository was forked to your personal space:

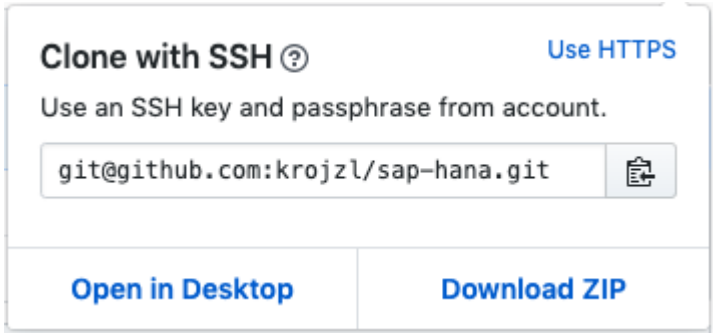


Detailed instructions are here: https://help.github.com/en/articles/fork-a-repo

1.4. Clone the forked repository

Clone your personal copy of the repository to your workstation.

1. Navigate to the forked repository: `https://github.com/<YOUR-USER>/sap-hana`
2. Click `Clone or download` button in upper-right corner of the page



   Note: In order to use SSH to connect to GitHub you need to click on `Use SSH` in upper-right corner of the panel

3. Copy the URL from the panel: `git@github.com:<YOUR-USER>/sap-hana.git`
4. Open Terminal and change directory to desired location
5. Run `git` command to clone the repository

```
# git clone git@github.com:<YOUR-USER>/sap-hana.git
Cloning into 'sap-hana'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
```

```
remote: Total 7 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
```

Detailed instructions are here: https://help.github.com/en/articles/fork-a-repo

1.5. Configure synchronization

Configure forked repository synchronization with main project repository.

1. Open Terminal and change directory to location of your local copy of the repository: `cd /path-to-your-repository/sap-hana`
2. List currently defined remote repositories:

```
# git remote -v
origin    git@github.com:<YOUR-USER>/sap-hana.git (fetch)
origin    git@github.com:<YOUR-USER>/sap-hana.git (push)
```

3. Add link to main project repository: `git remote add upstream git@github.com:sap-architecture/sap-hana.git`
4. List again defined remote repositories:

```
# git remote -v
origin    git@github.com:<YOUR-USER>/sap-hana.git (fetch)
origin    git@github.com:<YOUR-USER>/sap-hana.git (push)
upstream  git@github.com:sap-architecture/sap-hana.git (fetch)
upstream  git@github.com:sap-architecture/sap-hana.git (push)
```

Detailed instructions are here: https://help.github.com/en/articles/fork-a-repo

2. Recurrent synchronization

Perform following section on regular basis and always before you create Pull Request to main project.

2.1. Download new content from main project repository

Download and merge new updates from main project repository into your local repository on your workstation.

1. Open Terminal and change directory to location of your local copy of the repository: `cd /path-to-your-repository/sap-hana`
2. Fetch the branches and their content from the main project repository:

```
# git fetch upstream
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
```

```
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 0), reused 10 (delta 0), pack-reused 0
Unpacking objects: 100% (11/11), done.
From github.com:sap-architecture/sap-hana
* [new branch]      master     -> upstream/master
```

3. Change local repository branch to `master`

```
# git checkout master
Already on 'master'
Your branch is up to date with 'origin/master'.
```

4. Merge changes from `upstream/master` into your local `master` branch

```
# git merge upstream/master
Updating 11569ab..469f8ab
Fast-forward
CHANGELOG.md                    |   5 +++++
CONTRIBUTING.md                 |  46
++++++++++++++++++++++++++++++++++++++++++++
README.md                       |   3 ++-
images/contributing-clone-1.png | Bin 0 -> 18798 bytes
images/contributing-fork-1.png  | Bin 0 -> 6350 bytes
images/contributing-fork-2.png  | Bin 0 -> 15140 bytes
images/contributing-fork-3.png  | Bin 0 -> 12140 bytes
7 files changed, 53 insertions(+), 1 deletion(-)
create mode 100644 CHANGELOG.md
create mode 100644 CONTRIBUTING.md
create mode 100644 images/contributing-clone-1.png
create mode 100644 images/contributing-fork-1.png
create mode 100644 images/contributing-fork-2.png
create mode 100644 images/contributing-fork-3.png
```

5. In case there were merge conflicts you need to resolve them as described in https://help.github.com/en/articles/resolving-a-merge-conflict-using-the-command-line

Detailed instructions are here: https://help.github.com/en/articles/syncing-a-fork

2.2. Upload merged content to your GitHub repository

Upload merged content from your local repository on your workstation to your GitHub repository.

1. Open Terminal and change directory to location of your local copy of the repository: `cd /path-to-your-repository/sap-hana`

2. Upload the merged content in active branch to your GitHub repository

```
```

```
# git push
Total 0 (delta 0), reused 0 (delta 0)
To github.com:<YOUR-USER>/sap-hana.git
  11569ab..469f8ab  master -> master
```

3. Now you can see the content online in your own personal copy of the repository in GitHub: `https://github.com/<YOUR-USER>/sap-hana`

Detailed instructions are here: https://help.github.com/en/articles/pushing-commits-to-a-remote-repository

3. Add new Content and Commit

Deliver new content by editing files on your local workstation, push to online GitHub and review.

3.1. Add or edit the documentation

The documentation is written in Markdown language.

Additional information about Markdown:

- Basic writing and formatting syntax
- Working with advanced formatting
- Mastering Markdown
- GitHub Flavored Markdown Spec

Images should be uploaded into subdirectory `images` below directory where the page referencing the image is located.

Files should be uploaded into subdirectory `files` below directory where the page referencing the image is located.

Make sure you name all images and files in clear way.

All links to pages, images or files located in this repository should be based on relative paths - avoid using absolute paths as this will break the link functionality in forked repositories.

Use your favorite editor to add new content. Use Linting function to deliver clean and well-structured documentation.

Recommended editors:

- Visual Studio Code
  - plugin Markdown All in One
  - plugin Markdown Lint
  - plugin Auto Markdown TOC
- Atom Editor

3.2. Commit the content to your local repository

When unit of work is completed commit the changes to your local repository on your local workstation.

Use either your editor to commit the changes (recommended) or perform following commands:

1. Open Terminal and change directory to location of your local copy of the repository: `cd /path-to-your-repository/sap-hana`
2. Check the status

```
# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working
directory)

      modified:   CONTRIBUTING.md

no changes added to commit (use "git add" and/or "git commit -a")
```

   Note: In example above we can see file `CONTRIBUTING.md` is NOT part of commit.

3. Add files that should be included in commit
   - Add individual files: `git add <file_name>`
   - Add directory: `git add <directory_name>`

   In order to add all files, perform `git add .`

4. Check the status again

```
# git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

      modified:   CONTRIBUTING.md
```

5. Commit changes to active branch

```
# git commit -m "Update to CONTRIBUTING.md"
[master d6a2568] Update to CONTRIBUTING.md
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Detailed instructions are here: https://help.github.com/en/articles/adding-a-file-to-a-repository-using-the-command-line

3.3. Push changes to GitHub

Push your local changes from your local repository on your workstation to your GitHub repository.

1. Open Terminal and change directory to location of your local copy of the repository: `cd /path-to-your-repository/sap-hana`
2. Push changes from active branch to your GitHub repository

```
# git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 3.23 KiB | 3.23 MiB/s, done.
Total 6 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:<YOUR-USER>/sap-hana.git
   469f8ab..d6a2568  master -> master
```

3. Review your changes online in your own personal copy of the repository in GitHub: `https://github.com/<YOUR-USER>/sap-hana`

Detailed instructions are here: https://help.github.com/en/articles/adding-a-file-to-a-repository-using-the-command-line

4. Upload to main project repository

Create Pull Request (PR) from your own GitHub repository against main project repository.

4.1. Synchronize your content with main project

Very likely your updates took some time. Meanwhile the content in main project repository changed and might be out of sync with your own repository.

Follow procedure described in section 2. Recurrent synchronization.

4.2. Update `CHANGELOG.md` file

Make sure you document what was changed in `CHANGELOG.md` file. Make sure this very last change to avoid merge conflicts.

Follow procedure described in section 3. Add new Content and Commit to adjust the file.

4.3. Create Pull Request

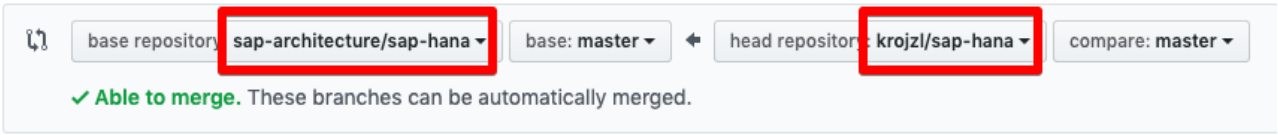
Before you create Pull Request (PR) make sure that:

- you downloaded and merged latest content from main project repository
- you resolved all merge conflicts
- you pushed all changes from your local repository on your workstation to online GitHub repository
- you reviewed your content online in GitHub `https://github.com/<YOUR-USER>/sap-hana` and confirmed the content is rendering correctly - that includes:
    - all images are properly displayed (and are relative)
    - all links are working properly (and are relative is pointing to page in same repository)

1. Navigate to the forked repository: `https://github.com/<YOUR-USER>/sap-hana`

2. Click `New pull request` button in upper-left part of the page

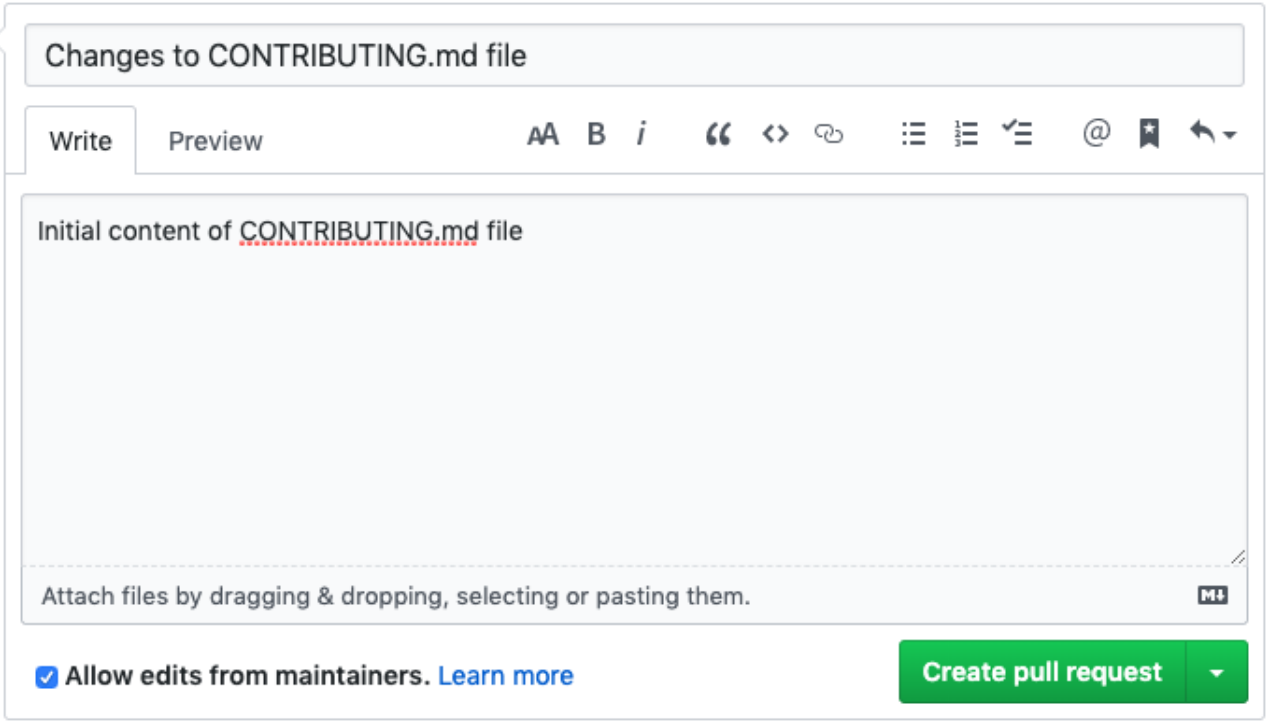
3. Review the Pull Request (PR) details



Note: On left side you see destination (main project repository) and on left side you see source (your own forked repository).

4. In case all visualized changes are ok press `Create pull request` button

5. Add title and meaningful description and click `Create pull request` button



6. Wait for your Pull Request to be reviewed and respond to any request for changes

   Note: Until your Pull Request is accepted all your commits that are pushed to your forked repository in GitHub will be automatically included in given Pull Request.

   If you want to make commits that are not included in Pull Request, consider using separate branch for your updates.

Detailed instructions are here: https://help.github.com/en/articles/creating-a-pull-request-from-a-fork

Requirements

Following requirements were taken into account for this Reference Architecture. Different requirements might lead to different Reference Architectures.

REQ1: Scale-out Driven Architecture

SAP HANA database is having ability to run in distributed way across multiple VMs (or physical servers). The Reference Architecture should take into account scale-out deployment option to ensure that single-node design can be easily extended into scale-out without major architectural redesign.

REQ2: High Availability (HA) Across Availability Zones (AZs)

Most IaaS offerings are supporting concept of Availability Zones - these are physically separated infrastructure segments that should not share any Single Point of Failure (SPOF), however are in close proximity to deliver low latencies required for Synchronous Replication. The Reference Architecture should take advantage of concept of Availability Zones to maximize the resiliency.

REQ3: Disaster Recovery (DR) into Remote Location

Due to close proximity of Availability Zones these might not be seen as sufficient for Disaster Recovery purpose. Therefore, the Reference Architecture should support Disaster Recovery function by shipping the data into Remote Location.

REQ4: SAP HANA Multitenant Database Containers (MDC) Implementation

SAP HANA is capable to run multiple independent database containers as part of one SAP HANA system. The Reference Architecture should support ability to run more than one database container.

REQ5: Virtual IP and Hostname for SAP HANA System Relocation

To enable SAP HANA portability the SAP HANA instance should be decoupled from underlying Operating System by using its own Virtual IP and Virtual Hostname. This will simplify the SAP HANA relocation and will prevent the need to change hostnames or IPs during the migrations. The Reference Architecture should support this capability.

REQ6: Support for Data Tiering Options (related to REQ2 and REQ4)

SAP invented various options how to distribute the data based on frequency of usage. The Reference Architecture should support following Data Tiering options (if technically viable):

- SAP HANA Native Storage Extensions (NSE)
- SAP HANA Extension Nodes
- SAP HANA Dynamic Tiering (DT)

Note: This requirement might be potentially conflicting with other requirements (in particular with REQ2 and REQ4).

REQ7: Enabled for "Instance move" (related to REQ5)

In certain cases, it might be required to move instance of SAP HANA database to new VM (for example move from VM to Physical Server). The Reference Architecture should support such relocation without the need to change any configuration on connecting applications by ensuring that IP address and Hostname will be preserved.

Note: This requirement is related to requirement REQ5.

REQ8: Enabled for "Tenant move" (related to REQ2 and REQ5)

SAP HANA database is supporting ability to relocate the database tenant into another SAP HANA database. The Reference Architecture should support such tenant relocation without the need to change any configuration on connecting applications by ensuring that IP address and Hostname will be preserved.

Note: This requirement is related to requirement REQ2 and REQ5.

REQ9: Fully TLS enabled (related to REQ4, REQ7 and REQ8)

SAP HANA is supporting ability to encrypt the database communication by using Transport Layer Secure (TLS) / Secure Sockets Layer (SSL) protocol. Since Fully Qualified Domain Name (FQDN) is part of the TLS/SSL configuration the Reference Architecture should properly define usage of FQDNs for individual database containers (related to REQ4) and minimize the need to recreate the certificates as result of "Instance Move" (REQ7) and/or "Tenant Move" (REQ8).

Architectural Decisions

Following Architectural Decisions (ADs) were made as part of this Reference Architecture. Different ADs might lead to different Reference Architectures.

AD1: High Availability Concept

ID	AD1
Name	High Availability Concept
Description	SAP HANA is supporting multiple fundamentally different High Availability concepts. One needs to be selected.
Assumptions	Overall design should support both single-node and scale-out in parallel next to each other. Objective is to minimize the Recovery Time Objective (RTO).
Options	1. SAP HANA Host Auto-Failover (HAF) 2. SAP HANA System Replication (synchronous)
Decision	2. SAP HANA System Replication (synchronous)
Justification	- this is the only option that is supporting REQ2 for scale-out systems - the Recovery Time Objective (RTO) values are significantly smaller compared to 1. - this option is supporting additional features like Active/Active (Read Enabled) or Secondary Time Travel
Comment	Recommended Replication Mode is `SYNC` in case there is possible shared Single Point of Failure (SPOF) or `SYNCMEM` in case of two physically separated infrastructures. Recommended Operation Mode is `logreplay` (or `logreplay_readaccess`).

AD2: Disaster Recovery Concept

ID	AD2
Name	Disaster Recovery Concept
Description	SAP HANA is supporting multiple fundamentally different Disaster Recovery concepts. One needs to be selected.
Assumptions	-
Options	1. Storage Replication 2. SAP HANA System Replication (asynchronous)
Decision	2. SAP HANA System Replication (asynchronous)
Justification	- option 1. might or might not be available and is unlikely to work cross-platform - option 2. is part of the product and therefore always available, it is platform independent and will work even cross-platform - as part of option 2. all data pages are checked for consistency during the transfer to secondary site

ID	AD2
Comment	Replication Mode must be `ASYNC` to avoid performance impact. Operation Mode must be same for all tiers (either `delta_datashipping` or `logreplay`/`logreplay_readaccess`), combining Operations Modes is not supported. Operation mode `logreplay_readaccess` is available only between primary and secondary system.

AD3: High Availability Takeover Automation

ID	AD3
Name	High Availability Takeover Automation
Description	There are different options/products how High Availability Takeover can be executed. One needs to be selected.
Assumptions	Objective is to minimize the Recovery Time Objective (RTO).
Options	1. Manual Takeover (no automation) 2. Pacemaker Cluster (Linux native solution) 3. 3rd Party Clustering Solution
Decision	2. Pacemaker Cluster (Linux native solution)
Justification	- option 1. is not satisfying requirement to minimize the Recovery Time Objective (RTO) value - option 2. is seen as recommended option by both OS vendor and SAP and is most common HA solution - option 3. might not be available across all platforms
Comment	Take into account specific Implementation Guidelines for each Infrastructure Platform.

Overall Architecture and Modularity

This section is explaining overall concept and how to use this Reference Architecture.

Modular Concept

Overall approach how this Reference Architecture was created is briefly described here.

Individual customers are having different requirements. These requirements are dictating how the resulting architecture will look like. Unfortunately, initial requirements are quite often not final - customers will need to enable new functions and features and this might subsequently drive the need to significantly change the architecture and reimplement SAP HANA database.

This Reference Architecture is trying to minimize the changed to SAP HANA architecture by offering individual "modules" which are pre-integrated together. These modules are optional and solution architects should choose only those modules which are required in given scenario. The benefit is that additional modules can be enabled later without the need to redeploy the SAP HANA database.

Resulting Architecture Is Too Complex

If all modules are selected, then resulting Reference Architecture is quite complex. This is because of complex requirements which are to be satisfied. If you consider the resulting architecture to be way too complicated, then try removing some optional modules or use simplified version of the modules (for example in area of Pacemaker configuration).

How to Read This Reference Architecture

Next sections are started by describing the basic setup - simple Single-Node and Scale-Out deployment in single Availability Zone. Subsequent modules are then always building up on top of each other - each module describing how to include one additional function until we end up with full scenario having all options enabled.

Since all modules are optional - it is technically impossible to document all permutations of choices as separate diagrams. Therefore, it is assumed that Solution Architect working with this Reference Architecture is familiar with SAP HANA technology and is able to properly combine available modules creating his own architecture based on preferred choices.

Module: Basic Architecture

This is foundational module for SAP HANA Reference Architecture. Two basic SAP HANA deployment options are explained - Single-Node and Scale-Out.

SAP HANA Multitenant Database Containers (MDC) vs Single-tenant Implementation

SAP HANA System can be implemented using one of two basic deployment types:

- SAP HANA Multitenant Database Containers
- SAP HANA Single-tenant Implementation

SAP HANA Multitenant Database Containers (MDC) were introduced in SAP HANA 1.0 SP09 and as of SAP HANA 2.0 SP01 the MDC concept is the only supported deployment option. Therefore, single-tenant implementation is considered outdated and is not recommended to be used.

This Reference Architecture is based on using Multitenant Database Containers deployment option however, it could be modified also for Single-tenant Implementation.

SAP HANA Stacking Options (MCOD, MCOS, MDC)

There are different techniques how to stack multiple SAP HANA Databases on same Virtual Machine (VM):

- Multiple Components on One Database (MCOD)
- Multiple Components on One System (MCOS)
- Multitenant Database Containers (MDC)

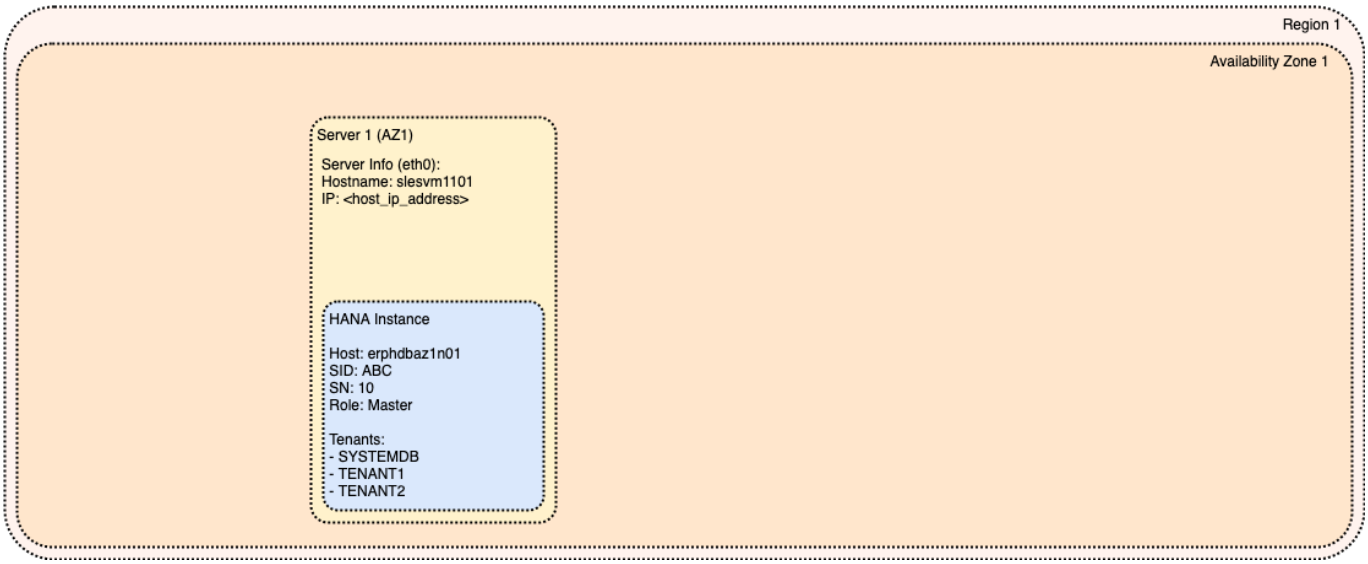
Recommended stacking option is MDC which is fully compatible with this Reference Architecture.

Stacking option MCOD is intended to be used only in specific scenarios wherever explicitly recommended by SAP.

Stacking option MCOS is not recommended and is not aligned with this Reference Architecture.

Additional information is available in section SAP HANA: Stacking Options (MCOD, MCOS, MDC) {TODO}.

Single-Node SAP HANA System (in single Availability Zone)



This is the simplest and most traditional deployment option. All components of SAP HANA database are running together on one single Virtual Machine (VM) having one Operating System (OS).

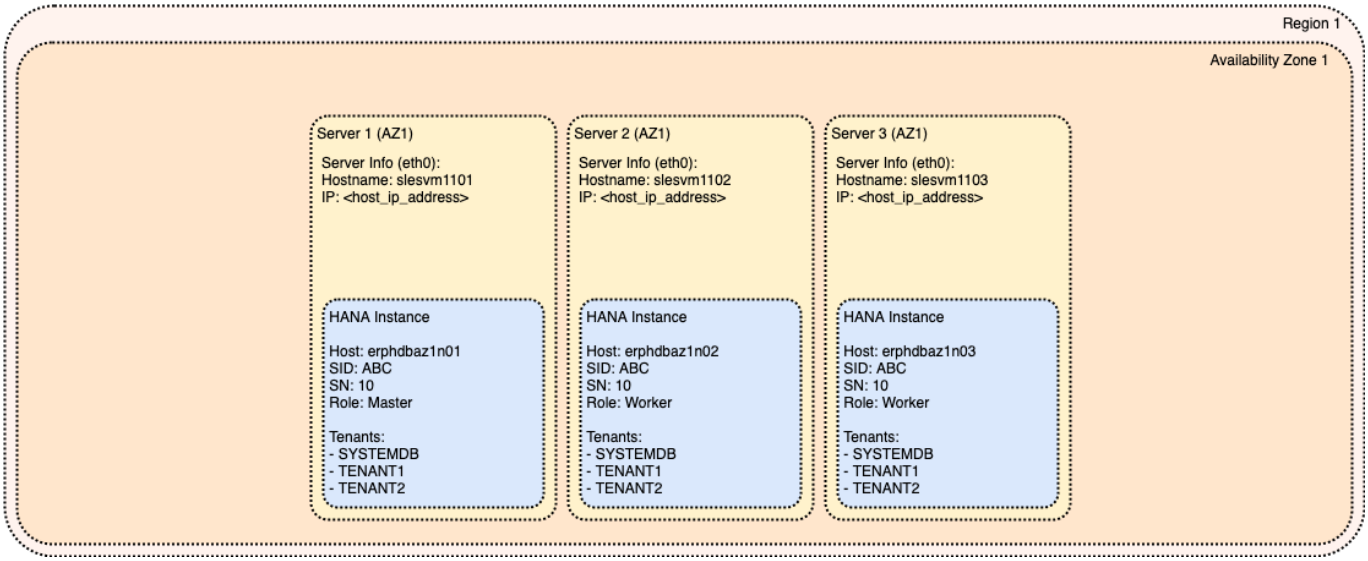
This deployment option is vulnerable to all failure scenarios:

- Regional failure (Disaster Event)
- Availability Zone failure
- VM failure / OS failure
- SAP HANA instance failure

Additional information:

- Master Guide: Single-Host System

Scale-Out SAP HANA System (in single Availability Zone)



This deployment option is running one SAP HANA database as multiple components distributed across multiple Virtual Machines (VMs) each having its own Operating System (OS).

SAP HANA internal architecture is based on "shared nothing" principle to allow almost linear scalability - this means that each instance (set of processes running on one Operating System) is having its own Data Files and Log Files which are NOT accessible by other instances (running on other VMs).

It is always recommended to prefer Single-Node deployment option over Scale-Out due to a performance reason. Scale-Out option is positioned to overcome the limitations of single VM (or physical server) in terms of maximal memory or CPU power.

This deployment option is not available for all SAP products - for more information check Master Guide and Installation Guide for given SAP product.

This Reference Architecture will use Scale-Out deployment option as base for all other modules. Single-Node architecture is seen as simplified version of Scale-Out deployment option. Any potential differences between Single-Node and Scale-Out designs will be explicitly highlighted.

This deployment option is vulnerable to all failure scenarios:

- Regional failure (Disaster Event)
- Availability Zone failure
- VM failure / OS failure
- SAP HANA instance failure

Note: Multiple nodes are NOT offering increased availability. Due to a "shared nothing" approach, failure on any of the nodes will impact whole SAP HANA database system.

Additional information:

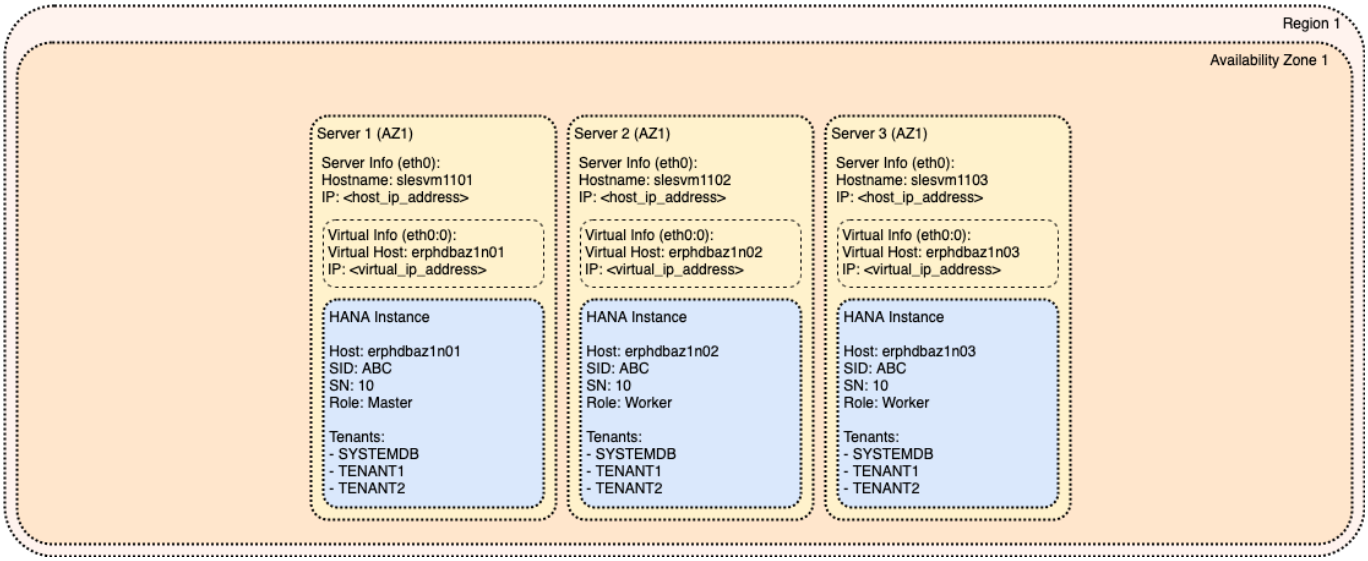
- Master Guide: Distributed System (Multiple-Host System)
- Administration Guide: Multiple-Host (Distributed) Systems

Module: Virtual Hostname/IP

In this module the basic architecture is extended by decoupling SAP HANA installation from OS installation by using Virtual Hostname and Virtual IP address dedicated to SAP HANA instance.

This module is prerequisite to support "Instance relocation" as documented here {TODO}.

Concept of Virtual Hostname and Virtual IP



Hostname and IP address is critical attribute uniquely identifying Operating System running on given Virtual Machine (VM). It is registered in many backend Systems of Management (including Active Directory integration, DNS integration, OS monitoring, etc.) and Systems of Records. Since is tightly coupled with the identity of given Operating System quite often it cannot be easily changed.

However, SAP HANA instance must be installed against some Hostname (see Administration Guide: Default Host Names and Virtual Host Names for additional information). Unless specific Hostname is provided then Hostname of Operating System is used.

Hostname used to install SAP HANA and its Fully Qualified Domain Name (FQDN) form is playing critical role in regard to connected applications and usage of Certificates to encrypt network communication (see Security Guide: TLS/SSL Configuration on the SAP HANA Server for additional information).

Therefore, many customers are having requirement to preserve the Hostname used during SAP HANA installation so that external connectivity is not disrupted following the migration.

Unfortunately, this is in direct contradiction with requirement to have different Hostname during SAP HANA System Replication which is recommended option how to minimize the business downtime during homogeneous migration of SAP HANA system (see Administration Guide: General Prerequisites for Configuring SAP HANA System Replication for additional information).

Convenient solution is usage of Virtual Hostname and Virtual IP for SAP HANA installation that is always following given SAP HANA instance. The advantage is that SAP HANA installation is decoupled from Operating System and can be easily relocated to new Operating System.

The procedure how to migrate SAP HANA system using Virtual Hostname/IP ("Instance relocation") is described here {TODO}.

The real implementation of Virtual Hostname is platform specific and is described in detail in Platform Specific Architecture part of the documentation.

Note the difference between the Virtual Hostname/IP and Cluster Hostname/IP. Purpose of Cluster Hostname/IP is to always to follow "active" instance of SAP HANA High Availability (HA) solution while Virtual Hostname/IP is static and is always following given instance regardless of its role in HA solution.

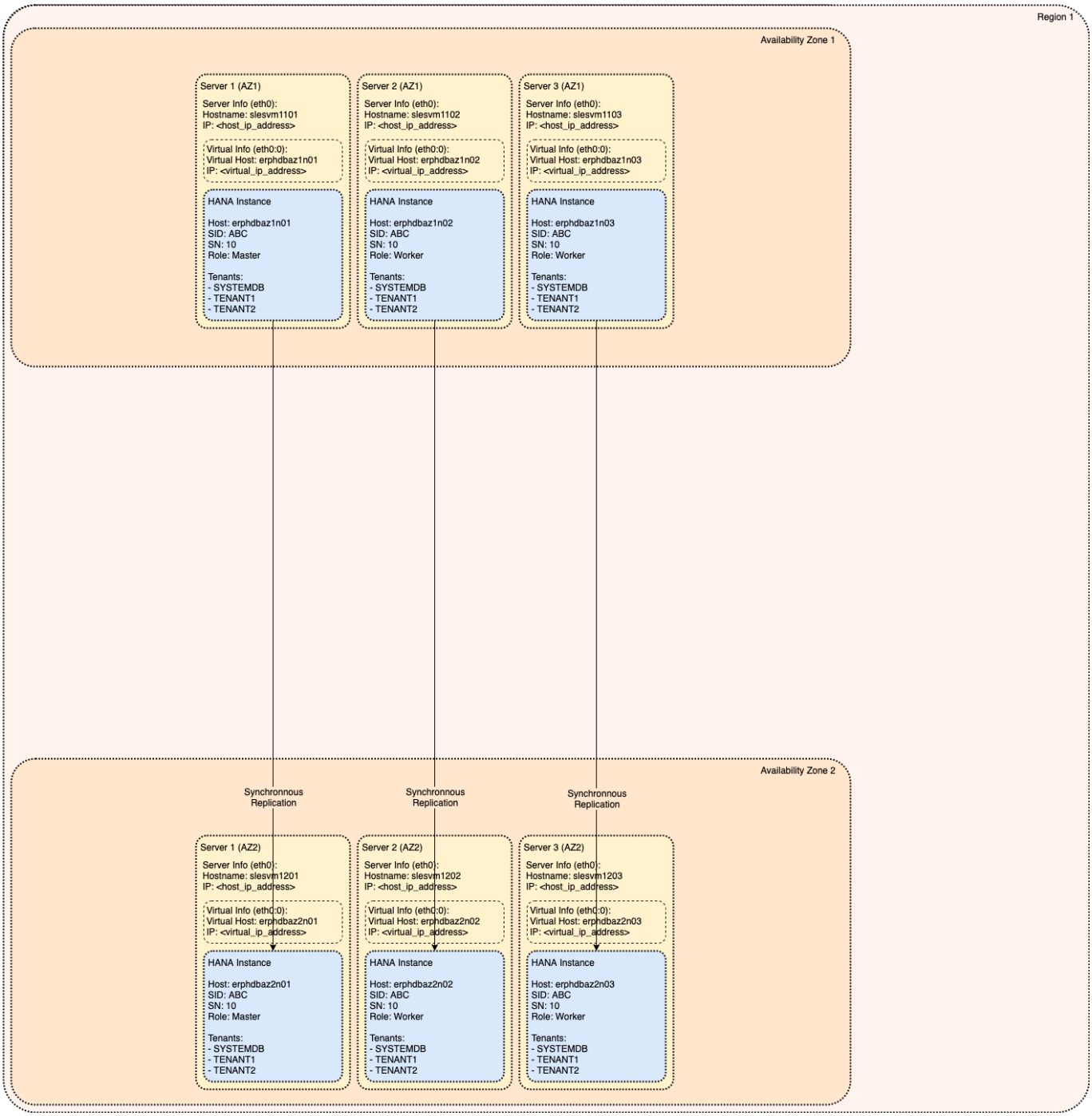
Module: High Availability

This module is enhancing the architecture by additional SAP HANA system increasing the Availability.

High Availability scenario is based on the Pacemaker cluster automating the takeover to secondary SAP HANA system.

Different options how Cluster IP can be configured are presented - each having its own advantages and disadvantages.

Enhanced Availability without Clustering (manually operated)



This is the very basic option how to increase SAP HANA Availability by adding secondary SAP HANA system in separate Availability Zone and configuring synchronous SAP HANA System Replication (see Administration Guide: Replication Modes for SAP HANA System Replication for additional information).

Following two Replication Modes are acceptable for Availability management:

- Synchronous on disk (`SYNC`)
- Synchronous in-memory (`SYNCMEM`)

Synchronous on disk (`SYNC`) Replication Mode is having higher latency impact because it waits for disk write operation on secondary SAP HANA system to complete. The advantage is that Recovery Point Objective (RPO) is guaranteed to be zero (no data loss possible as long as secondary system is connected). This option is recommended in situations where we have potential Single Point of Failure (SPOF) shared between both primary and secondary SAP HANA system.

Synchronous in-memory (`SYNCMEM`) Replication Mode is having Recovery Point Objective (RPO) only "close to zero" because information on secondary SAP HANA database is written to disk asynchronously. The advantage is improved performance because the latency impact is reduced by disk write operation. However this Replication Mode can lead to data loss in case that both primary and secondary SAP HANA system will fail at the same time - therefore it recommended only in scenarios where there is no Single Point of Failure (SPOF) shared between both primary and secondary system - for example in combination with Availability Zones.

Note that `Full Sync Option` as described in Administration Guide: Full Sync Option for SAP HANA System Replication is not suitable for any High Availability usage. This is because any failure (either of primary or secondary SAP HANA System) will result in remaining SAP HANA System to be blocked.

Asynchronous (`ASYNC`) Replication Mode is not acceptable because it cannot guarantee Recovery Point Objective (RPO) to be zero or "close to zero".

In this scenario the takeover is executed manually by SAP HANA administrator (see Administration Guide: Performing a Takeover for additional information) and therefore the Recovery Time Objective (RTO) depends mainly on monitoring lag and reaction time of support teams.

There are two techniques how to ensure that application connectivity to SAP HANA database is not disrupted following the takeover operation (see Administration Guide: Client Connection Recovery After Takeover for additional information):

- IP redirection (referred as Cluster IP)
- DNS redirection

IP redirection (repointing Cluster IP to new primary SAP HANA system after takeover) or DNS redirection is executed manually following the takeover action.

The implementation details for Cluster IP are platform specific and are described in Platform Specific Architecture part of the documentation.

High Availability with Pacemaker Cluster

In order to decrease the Recovery Time Objective (RTO) the takeover process must be automated. This can be done using Pacemaker as cluster management solution.

Additional Information:

- ClusterLabs: Pacemaker Documentation
- SLES12 SP4: High Availability Extension - Administration Guide
- SLES15 GA: High Availability Extension - Administration Guide
- RHEL7: High Availability Add-On Reference

When dealing with Pacemaker cluster there are two topics that are impacting the final architecture:

- Implementation of Cluster IP
- Fencing Mechanism

Technical implementation of Cluster IP is specific to given infrastructure and therefore is in detail described in Platform Specific Architecture part of the documentation. In this section of the Reference Architecture we will only discuss generic concepts how to design Cluster IP configuration.

Fencing is critical mechanism protecting data from being corrupted. What is fencing and how it works is explained here:

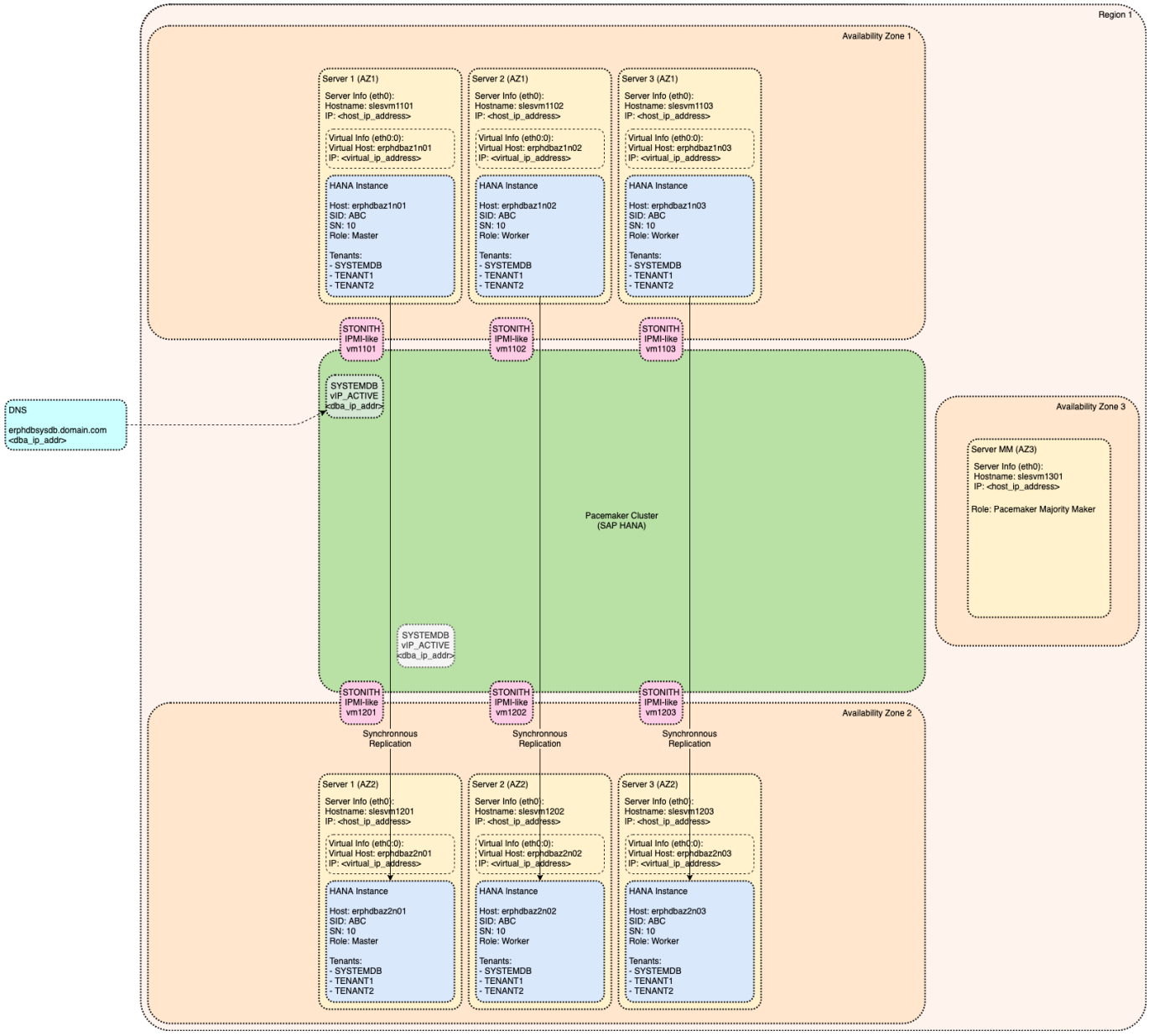
- Split-brain, Quorum, and Fencing
- ClusterLabs: What is Fencing?

Sections below are explaining two basic options how to implement Fencing Mechanism:

- IPMI-like Fencing
- SBD (Storage Based Death) Fencing

Recommendation which option to use on each platform is described in Platform Specific Architecture part of the documentation.

IPMI-like Fencing



IPMI-like fencing approach is based on direct access to Management Interface of given server which is also called IPMI (Intelligent Platform Management Interface) and which is having ability to power down the given server.

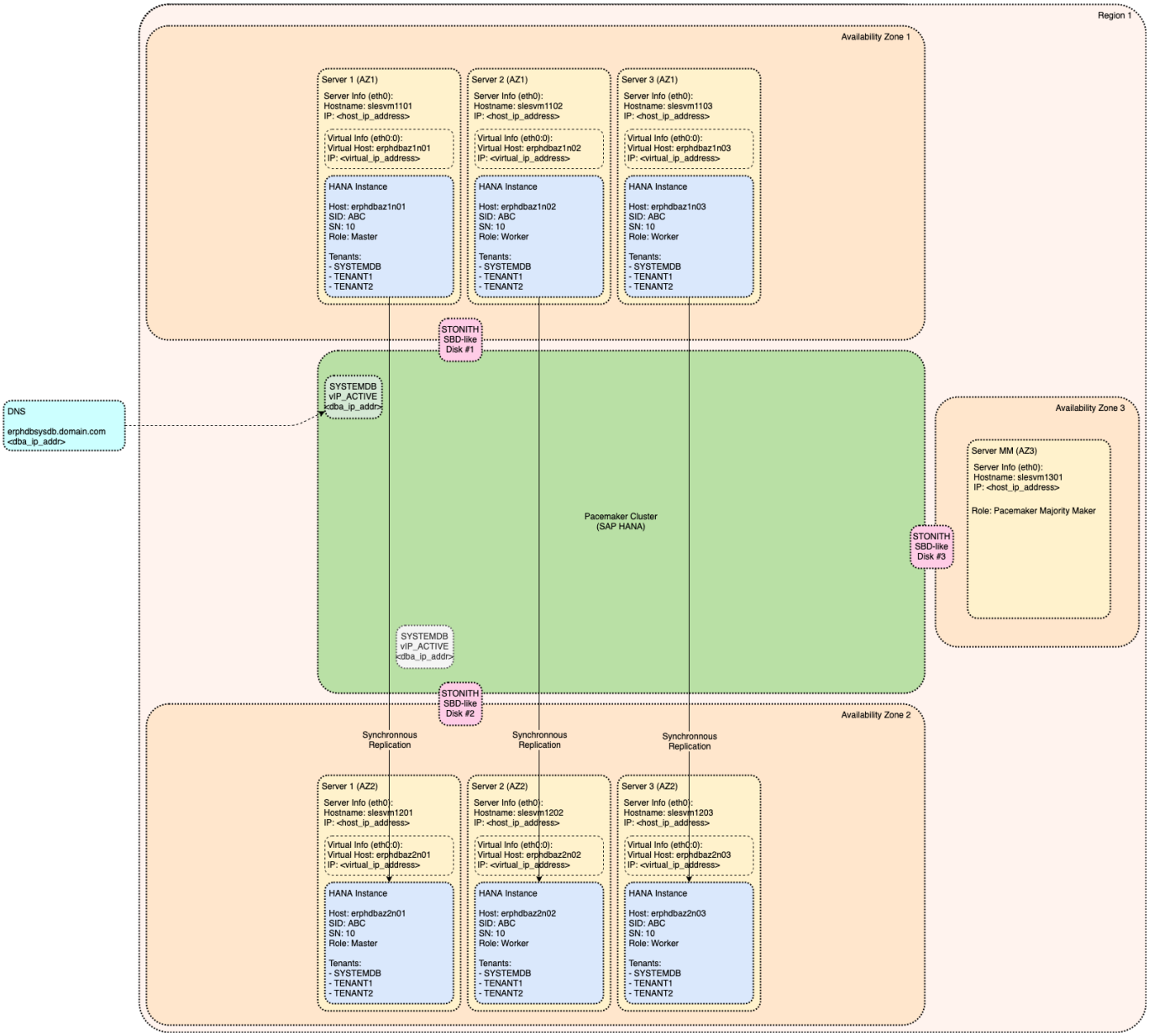
Here are some example implementations of IPMI-like agents:

- Amazon Web Services (AWS): external/ec2
- Microsoft Azure: fence_azure_arm
- Google Cloud Platform (GCP): external/gcpstonith
- On-premise Bare Metal: external/ipmi

All of these agents are having same purpose - to kill the Virtual Machine or Bare Metal Server as soon as technically possible. The goal is not to perform graceful shutdown but to immediately terminate the server to ensure that it is down before secondary server can takeover.

In cluster configuration each SAP HANA server needs to have its own IPMI-like fencing agent configured and fully operational. The IPMI-like fencing agent is always called from the remote side (for example secondary systems are fencing off primary systems).

SBD (Storage Based Death) Fencing



SBD (Storage Based Death) fencing is based on different approach.

The SBD device is shared raw disk (can be connected via Fibre Channel, Fibre Channel over Ethernet or iSCSI) that is used to send messages to other nodes.

When SBD device is initiated it will overwrite the beginning of the disk device with messaging slot structure. This structure is used by individual nodes to send messages to other nodes. Each cluster node is running SBD daemon that is watching the slot dedicated to given node and performing associated actions.

In case of fencing event the node that is triggering the fencing will write the "poison pill" message to the slot associated with target system. SBD daemon on target system is monitoring given slot and once it will read the "poison pill" message it will execute suicide action (self-fencing itself from the cluster by instant powering off).

Here is implementation of SBD agent:

- external/sbd

Recommended amount of SBD devices is either:

- Three SBD devices - each in separate Availability Zone (visualized on picture above)
- One SBD device - in 3rd Availability Zone (other than used by SAP HANA VMs)

Additional Information:

- Linux-HA: SBD Fencing
- sbd - STONITH Block Device daemon

Majority Maker Node

As said above Fencing concept is vital to protect the data from corruption. However, in case of communication issues between cluster nodes there is risk that individual nodes will continuously keep fencing each other by taking over the role of primary system.

Therefore, there is second equally important concept called Quorum that is deciding which subgroup of cluster nodes is entitled to become primary.

SAP HANA Scale-Out Scenario

Most simple implementation of Quorum is to use odd number of nodes and base the Quorum logic on majority of nodes in subgroup.

This method is applicable to SAP HANA Scale-Out configurations where we have same number of nodes on each Availability Zone and we need one additional VM in 3rd Availability Zone to act as "Majority Maker" helping to decide which side will become primary.

Recommended settings are described in SLES12 SP2: SAP HANA System Replication Scale-Out - Cluster Bootstrap and more.

SAP HANA Single-Node Scenario

Concept of using "Majority Maker" is also applicable to SAP HANA Single-Node implementations (two node clusters). However, such small clusters can be also implemented using special `two_node` Quorum approach as described in SLES12 SP4: High Availability Extension - Corosync Configuration for Two-Node Clusters:

```
quorum {
 provider: corosync_votequorum
 expected_votes: 2
 two_node: 1
}
```

Note that setting `two_node: 1` value will implicitly configure `wait_for_all: 1`.

The configuration is explained in `votequorum` man pages Man Pages: VOTEQUORUM(5) and in following document New quorum features in Corosync 2.

Effectively the configuration setting is adjusting how Quorum is calculated. When cluster is started for the first time (both nodes down) then `wait_for_all: 1` parameter is ensuring that both nodes need to be available to achieve the quorum. This is critical to protect the consistency of data as explained in following blogs:

- Be Prepared for Using Pacemaker Cluster for SAP HANA – Part 1: Basics
- Be Prepared for Using Pacemaker Cluster for SAP HANA – Part 2: Failure of Both Nodes

However, if the cluster is already active (and Quorum was achieved) then parameter `two_node: 1` will ensure that in case that one node will fail the other node is still having Quorum even if it is not having majority.

In case of split-brain situation (when both nodes are active however unable to communicate), both nodes will have Quorum and both nodes will race to fence the other node. The node that will win the race will be primary.

This option is applicable only to Single-Node scenario where we have one SAP HANA System in each Availability Zone. In such case "Majority Maker" VM in 3rd Availability Zone is not required (although possible).

Cluster IP Design

The simplest SAP HANA High Availability scenario needs only one Cluster IP that is following Active Nameserver of primary SAP HANA system (which is where System Database is available).

Each tenant in SAP HANA system is having its own ports that can be used to directly connect to given Tenant Database. Although this direct connection is possible it is recommended to connect indirectly by specifying the port for the System Database (`3xx13` for ODBC/JDBC/SQLDBC access) and the Tenant Database name. SAP HANA system will ensure that connection is internally rerouted to target Tenant Database.

In case of takeover event the Pacemaker cluster will ensure that Cluster IP is moved to new primary SAP HANA system.

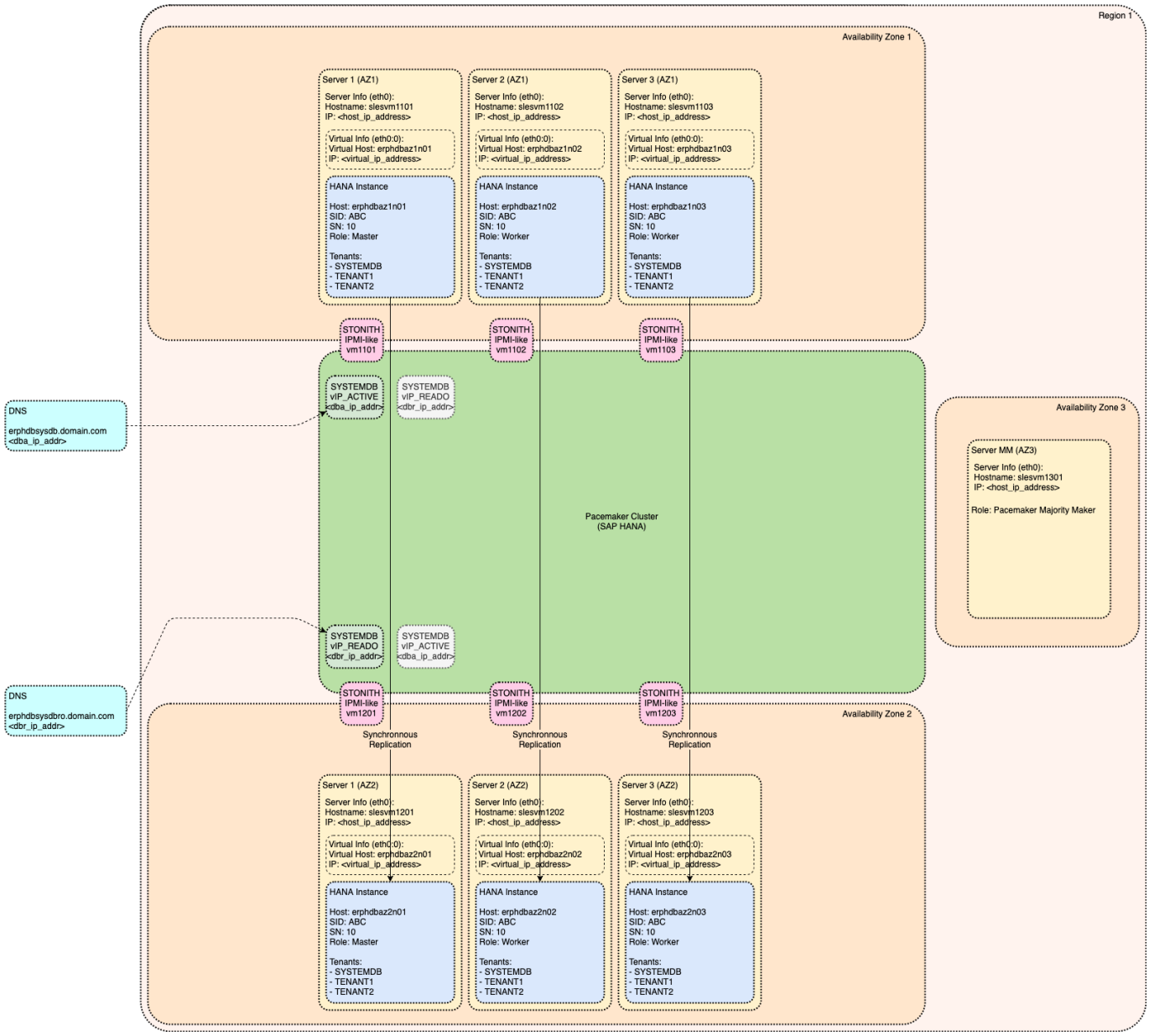
As explained above the technical implementation of Cluster IP is in detail covered in Platform Specific Architecture part of the documentation.

Additional Information:

- Administration Guide: Server Components of the SAP HANA Database
- Tenant Databases: Connections for Tenant Databases
- Tenant Databases: Scale-Out Architecture of Tenant Databases
- Administration Guide: Connections from Database Clients and Web Clients to SAP HANA
- Administration Guide: Connections for Distributed SAP HANA Systems

- TCP/IP Ports of All SAP Products

Active/Active High Availability with Pacemaker Cluster



Historically secondary SAP HANA System was closed, and connection attempts were rejected (this is still valid for Operation Modes `delta_datashipping` or `logreplay`).

Since SAP HANA 2.0 new operation mode `logreplay_readaccess` is available which is offering capability to open secondary SAP HANA System for read-only access.

As explained in SAP HANA Administration Guide: Connection Types and in Administration Guide: Virtual IP Address Handling secondary Cluster IP following Active Nameserver of secondary SAP HANA system is required.

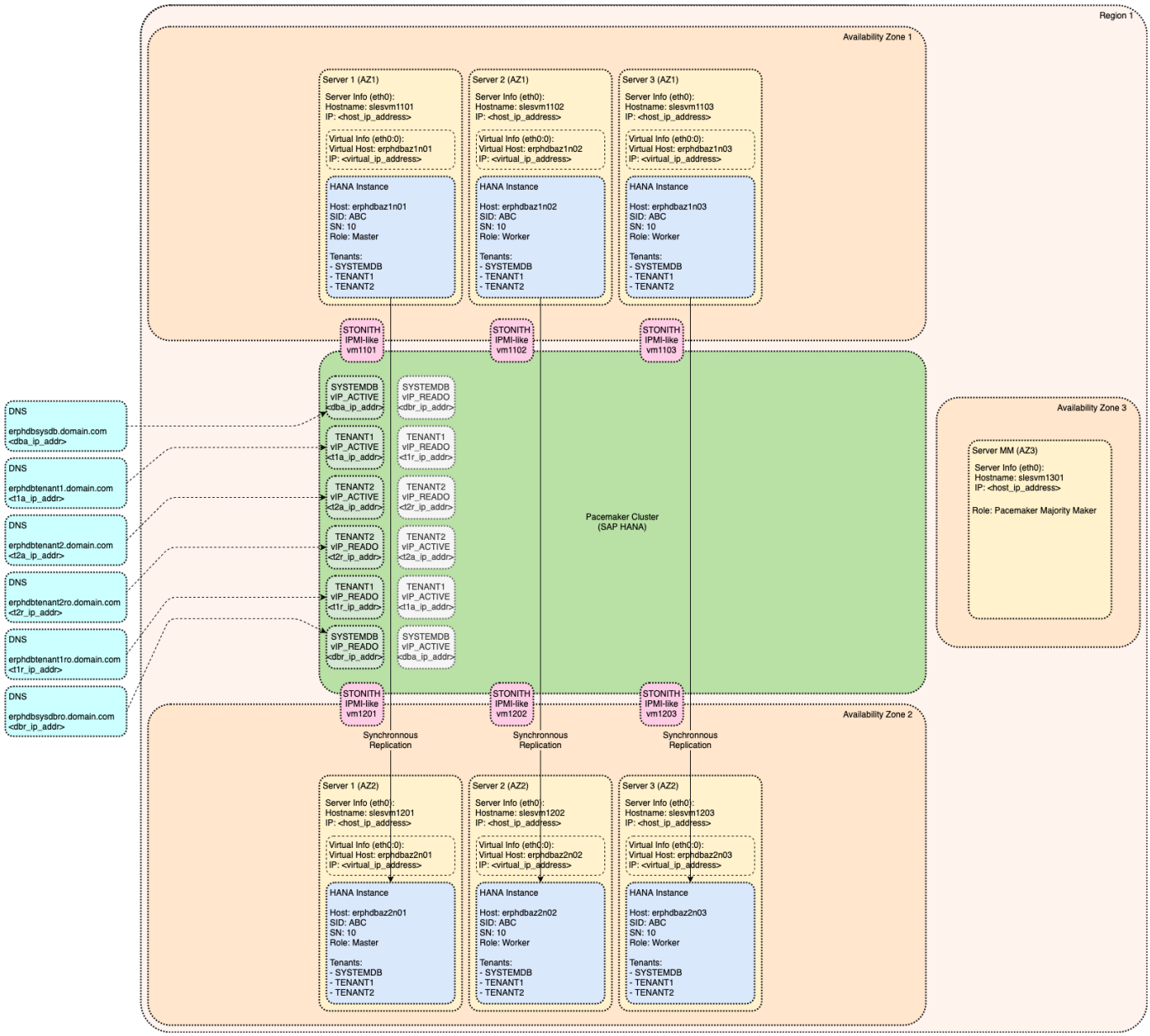
During normal operation both Cluster IP addresses are anti-collocated to each other - primary Cluster IP address is following primary SAP HANA System and secondary Cluster IP address is following secondary SAP HANA System.

As part of takeover event the Pacemaker cluster will switch location of both IP addresses along with change of primary and secondary roles of SAP HANA Systems.

Additional Information:

- Administration Guide: Active/Active (Read Enabled)
- Administration Guide: Connection Types
- Administration Guide: Virtual IP Address Handling

Active/Active High Availability with Pacemaker Cluster (enabled for "Tenant Move")



SAP HANA is offering option to move Tenant Database from existing SAP HANA System to new SAP HANA System having different `SID` and `system_number`.

Architecture documented in previous section is having one big limitation related to "Tenant Move" operation. The design is supporting multiple Tenant Databases on one SAP HANA cluster however, all tenants are accessed over one shared Cluster IP.

In such configuration when Tenant Database is moved, all applications connecting to this Tenant Database must be reconfigured to use Cluster IP of target SAP HANA cluster.

To make "Tenant Move" operation as seamless as possible each tenant needs to have its own Cluster IP that will be moved to target SAP HANA cluster along with given tenant.

All tenant-specific Cluster IPs are implemented in same way as System Database Cluster IP, they are following Active Nameserver of primary SAP HANA system - which is where System Database, used to connect to individual tenants, is available.

Second challenge that needs to be addressed is port used for connecting to System Database (`3xx13` for ODBC/JDBC/SQLDBC access). This port is dependent on `system_numer` of given SAP HANA System and therefore can differ. Solution to this problem is to allocate additional port (same across all SAP HANA Systems) on which System Database Tenant will listen. The procedure is described in Administration Guide: Configure Host-Independent Tenant Addresses.

The procedure how to move Tenant Database to new SAP HANA System ("Tenant relocation") is described here {TODO}.

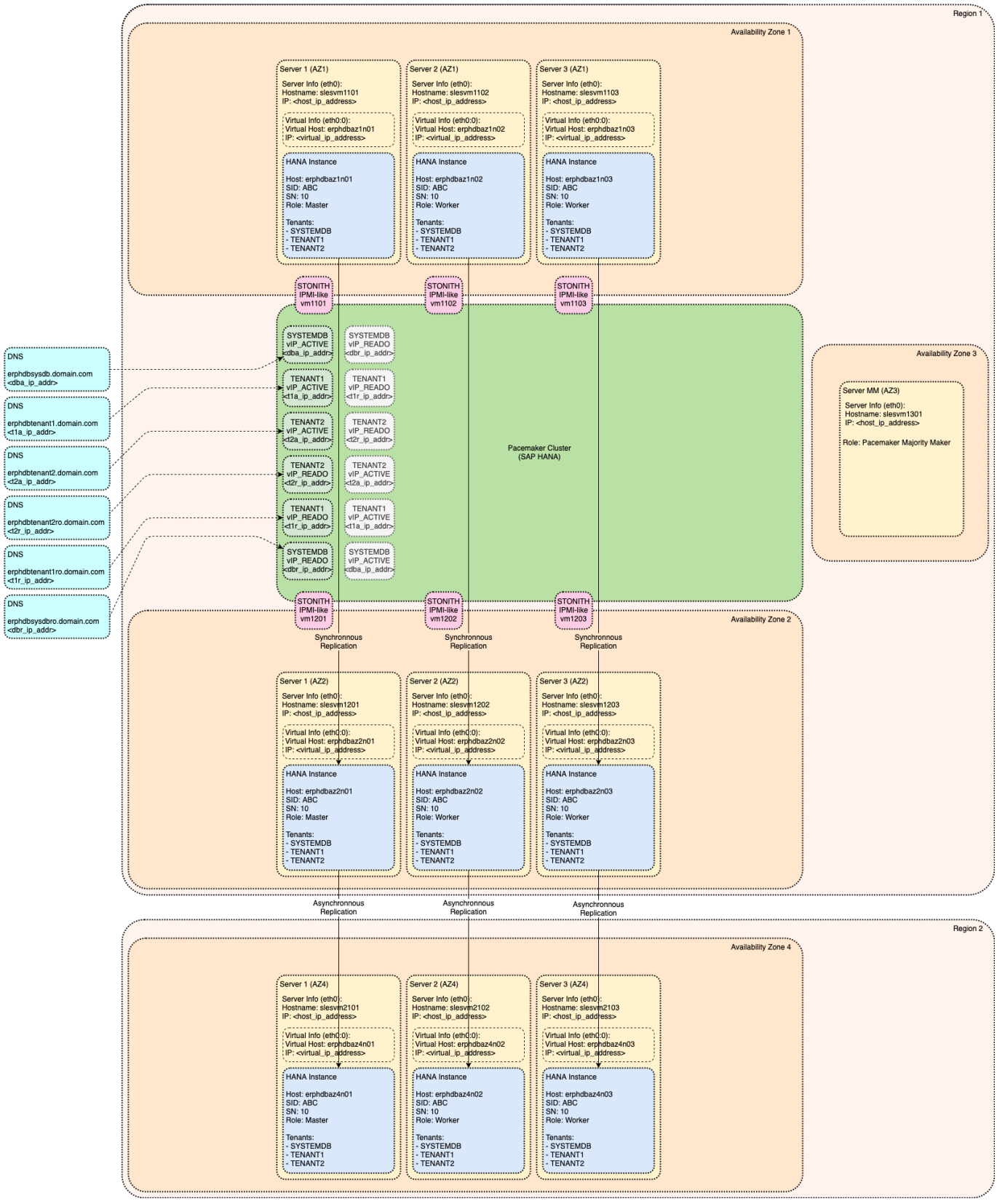
Additional Information:

- Administration Guide: Copying and Moving Tenant Databases
- Administration Guide: Configure Host-Independent Tenant Addresses

Module: Disaster Recovery

Disaster Recovery module is enhancing the design by adding protection against regional disaster. Such disaster events can be caused by nature (floods, tornadoes, hurricanes or earthquakes), by humans (strikes, terrorism) or by technology (power blackout).

Disaster Recovery Concept



All events mentioned above are typically having wide area of impact. Since High Availability design is based on synchronous replication, there are chances that both primary and secondary system will be impacted by regional disaster at the same time. Therefore, there is need to replicate the data outside the impacted area.

SAP HANA Asynchronous (ASYNC) Replication is recommended approach how to ship the data to Disaster Recovery location (independent region). The advantage is that Asynchronous Replication Mode is not susceptible to increased latency because the replication is happening on background.

As explained in System Replication Guide: Replication Performance Problems the network bandwidth is still critical even for Asynchronous Replication Mode.

Number of active nodes of the target SAP HANA System in the Disaster Recovery location must be same as on source system (see Administration Guide: General Prerequisites for Configuring SAP HANA System Replication for additional information).

Additional Information:

- System Replication Guide: SAP HANA System Replication
- Administration Guide: SAP HANA System Replication
- Administration Guide: SAP HANA Multitier System Replication
- Administration Guide: SAP HANA Multitarget System Replication
- Administration Guide: Disaster Recovery Scenarios for Multitarget System Replication

Module: Data Tiering Options

SAP if offering range of capabilities how to optimize costs by segregating data into different storage and processing tiers. This module is discussing how individual Data Tiering options can be implemented as part of this Reference Architecture.

- Module: Data Tiering Options
  - Overview of Data Tiering Options for SAP HANA
  - Persistent Memory (Non-Volatile Random Access Memory - NVRAM)
  - SAP HANA Native Storage Extensions (NSE)
  - SAP HANA Extension Nodes
  - SAP HANA Dynamic Tiering (DT)

Overview of Data Tiering Options for SAP HANA

SAP is dividing the data based on the aging characteristics of the data and frequency of usage. Following data temperature tiers and tiering options are available:

- Hot Data
  - Dynamic Random Access Memory (DRAM)
  - Persistent Memory (Non-Volatile Random Access Memory - NVRAM)
- Warm Data
  - Native Storage Extensions (NSE)
  - Extension Nodes
  - Dynamic Tiering (DT)
- Cold Data
  - SAP Data Hub / SAP Data Intelligence
  - SAP HANA Spark Controller (Hadoop)

Selected Data Tiering Options are discussed in sections below.

Additional Information:

- Administration Guide: Data Tiering

Persistent Memory (Non-Volatile Random Access Memory - NVRAM)

SAP HANA in-memory data can be divided into following usage types:

- Main Data fragments (the in-memory copy of tables; infrequent changes)
- Delta Data fragments (update information; frequent changes)
- Temporary Data fragments (computational data; very frequent changes)

Server memory must be combination of Traditional RAM (`DRAM`) and Persistent Memory (`NVRAM`). Traditional RAM (`DRAM`) is required during Operating System start and is also offering better performance for write operations. On the other hand Persistent RAM (`NVRAM`) is cheaper and bigger and almost as fast as `DRAM` for read operations.

Therefore, Persistent Memory is intended only for Main Data fragments of Column Store tables that are changed very infrequently (only during Delta Merge operation).

Persistent Memory is supported since SAP HANA 2.0 SP03 (revision 35 and higher) and SAP HANA 2.0 SP04.

Usage of Persistent Memory can be activated on the level of whole SAP HANA Database, selected Tables, selected Table Partitions or only selected Table Columns.

Although mixed combinations with primary systems having Persistent Memory and secondary systems without Persistent Memory and visa versa are supported, it is not recommended for High Availability purpose. In any case proper memory sizing must be ensured to avoid out-of-memory situations after takeover.

Additional Information:

- Administration Guide: The Delta Merge Operation
- Administration Guide: Persistent Memory
- SAP Note 2618154: SAP HANA Persistent Memory - Release Information
- SAP Note 2700084: FAQ: SAP HANA Persistent Memory

SAP HANA Native Storage Extensions (NSE)

SAP HANA is offering native option how to manage less frequently accessed data using built-in warm data store capability called Native Storage Extensions (NSE).

Data management of hot data is well described in Administration Guide: Memory Management in the Column Store.

Hot data is normally stored in "In-Memory Column Store". SAP HANA is automatically loading complete data structures (Table Columns or Table Column Partitions) into the memory based on first usage and will keep all data in memory as long as possible. These data structures are unloaded from memory only in case that allocated memory has reached the maximum limit and memory is required for processing other workload. In such case least recently used data structures are unloaded first.

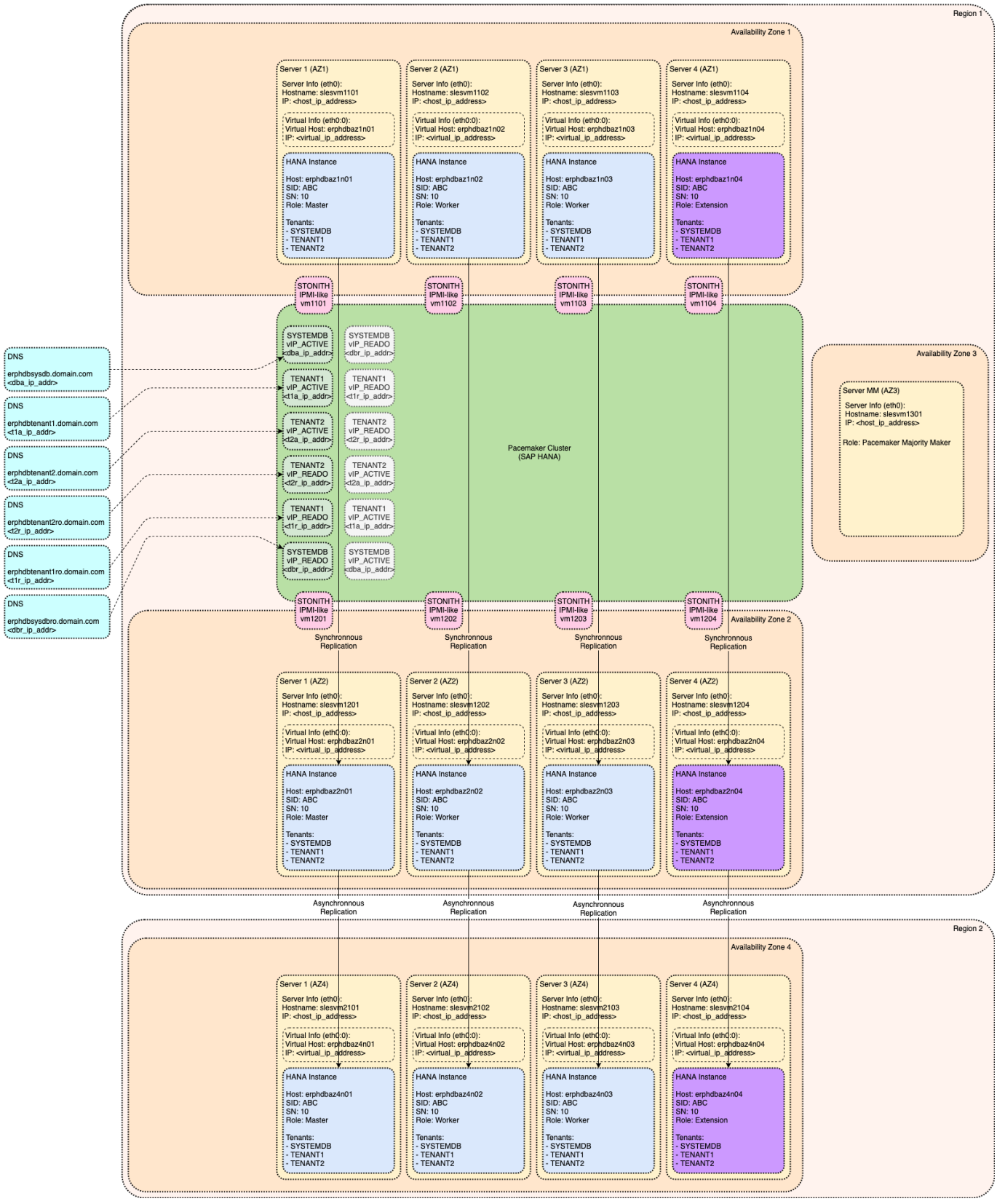
SAP HANA Native Storage Extensions functionality is offering different approach based on "Disk Based Column Store". It can be activated for selected database objects (tables, partitions or columns). The data structures (Table Columns or Table Column Partitions) are kept on disk and only selected data pages are loaded into memory into "Buffer Cache". This concept if well known from all other classical databases.

By using "Buffer Cache" that is significanly smaller than size of data in "Disk Based Column Store", Native Storage Extensions capability is increasing maximum amount of data that can be stored in SAP HANA database. Therefore, total storage requirements are also increased which needs to be reflected by infrastructure.

SAP HANA Native Storage Extension feature is supported since SAP HANA 2.0 SP04 and is limited only to Single-node SAP HANA Systems. Please note other functional restrictions as mentioned in SAP Note 2771956: SAP HANA Native Storage Extension Functional Restrictions.

- Administration Guide: SAP HANA Native Storage Extension
- SAP HANA Native Storage Extension Whitepaper
- SAP Note 2799997: FAQ: SAP HANA Native Storage Extension (NSE)

SAP HANA Extension Nodes



SAP HANA Scale-Out Systems can leverage SAP HANA Extension Nodes capability - new type of SAP HANA instance used exclusively for warm data.

SAP HANA Extension Node (configured as a slave node, worker group value `worker_dt`) is storing the warm data in "In-Memory Column Store" like regular SAP HANA node used for hot data. Since the warm data is less frequently used, the performance for `SELECT` statements against this data is not seen as important. Therefore, we can overload this node with amount of data to be doubled or in some cases even quadrupled.

Because the data is stored in "In-Memory Column Store" the internal mechanics is same as described in Administration Guide: Memory Management in the Column Store and in previous section. Due to a high volume of data in given node, the data structures are loaded and unloaded much more often than hot data in other nodes. Since this is associated with performance degradation, the Extension Node must be dedicated only for warm data.

Warm data must be placed in separate Tables or in separate Table Partitions. Subsequently those Tables and Table Partitions are relocated to SAP HANA Extension Node(s). Note that as described in Module: Basic Architecture each SAP HANA node is having its own subset of data in its own data files.

SAP HANA Extension Nodes are supported since SAP HANA 1.0 SP12 (for BW scenario) and since SAP HANA 2.0 SP03 (native scenario) and are limited only to Scale-Out SAP HANA Systems.

For SAP BW scenarios the hardware used for SAP HANA Extention Nodes can differ compared to other worker nodes starting from SAP HANA 2.0 SP03. For native scenarios this is supported from SAP HANA 2.0 SP4.

- Administration Guide: Extension Node
- Administration Guide: Redistributing Tables in a Scaleout SAP HANA System
- More Details – HANA Extension Nodes for BW-on-HANA

SAP HANA Dynamic Tiering (DT)

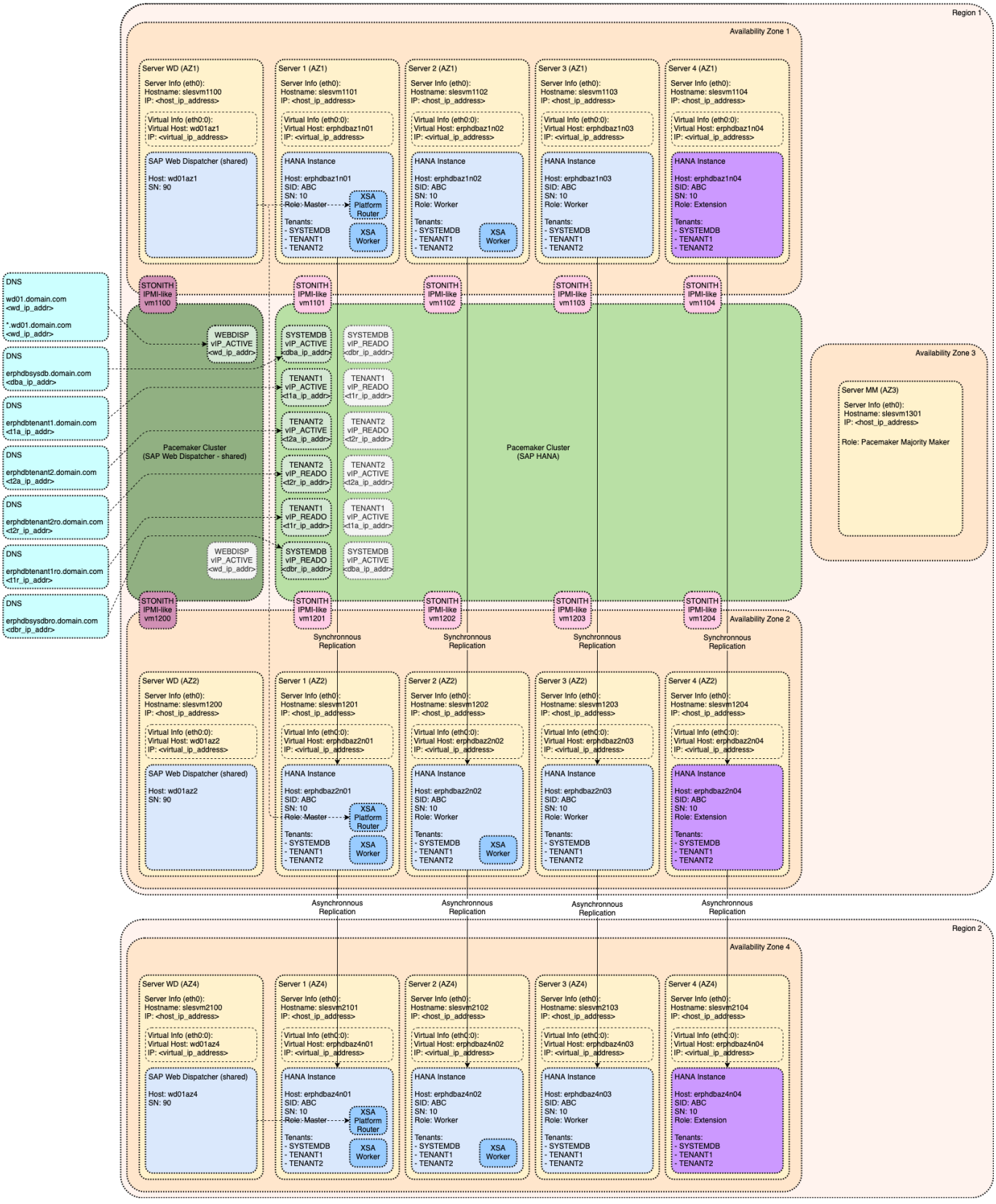
Not possible in combination with Pacemaker Clustering

https://help.sap.com/viewer/product/SAP_HANA_DYNAMIC_TIERING/2.00.04/en-US

Module: XSA (SAP HANA extended application services, advanced model)

Description

XSA (SAP HANA extended application services, advanced model)



Alternative Implementations

Description

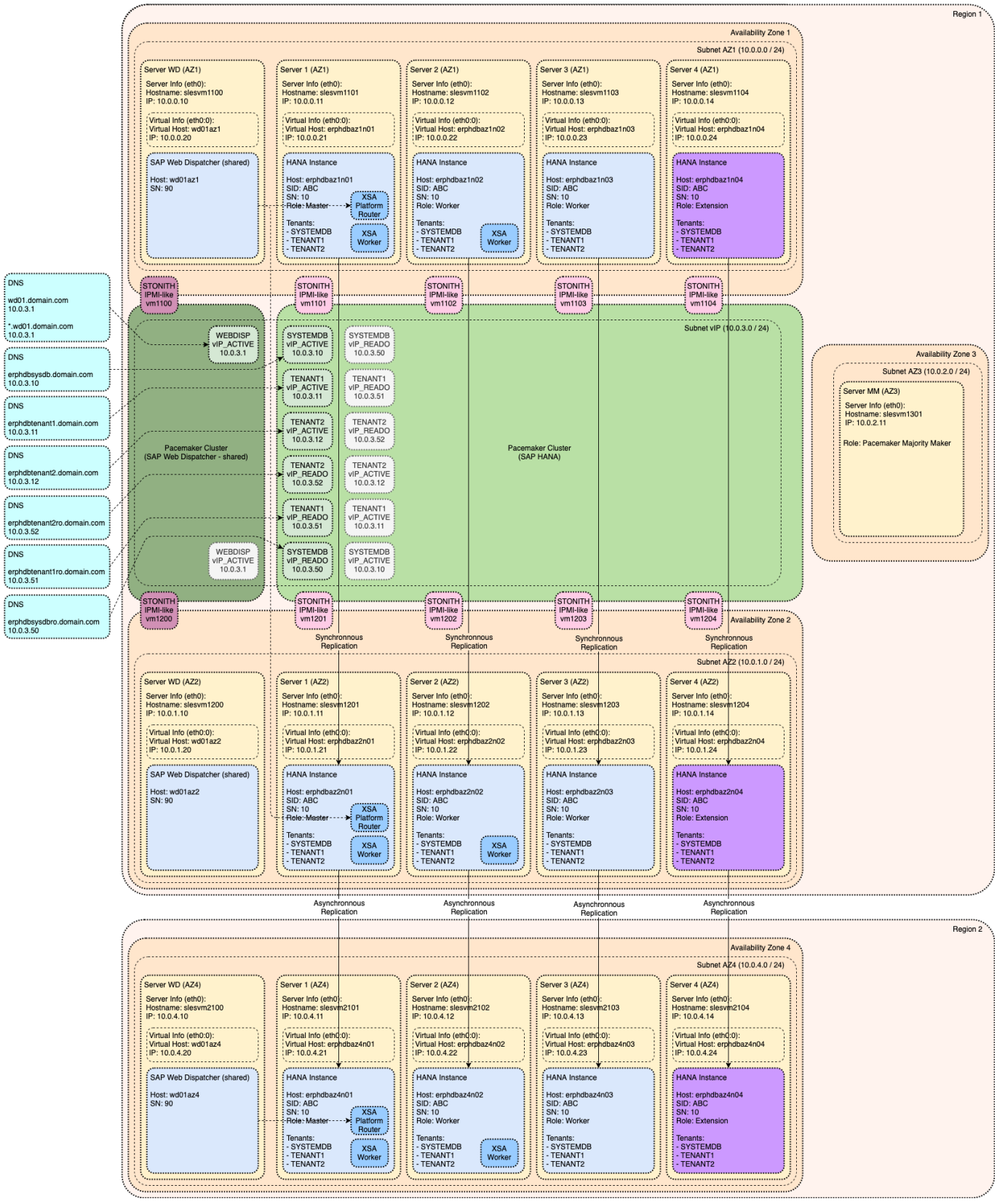
SAP HANA Host Auto-Failover (in single Availability Zone)

- single AZ implementation
- high cluster takeover times
- lower costs

Platform Specific Architecture for AWS (Amazon Web Services)

Description

AWS: Overall Architecture

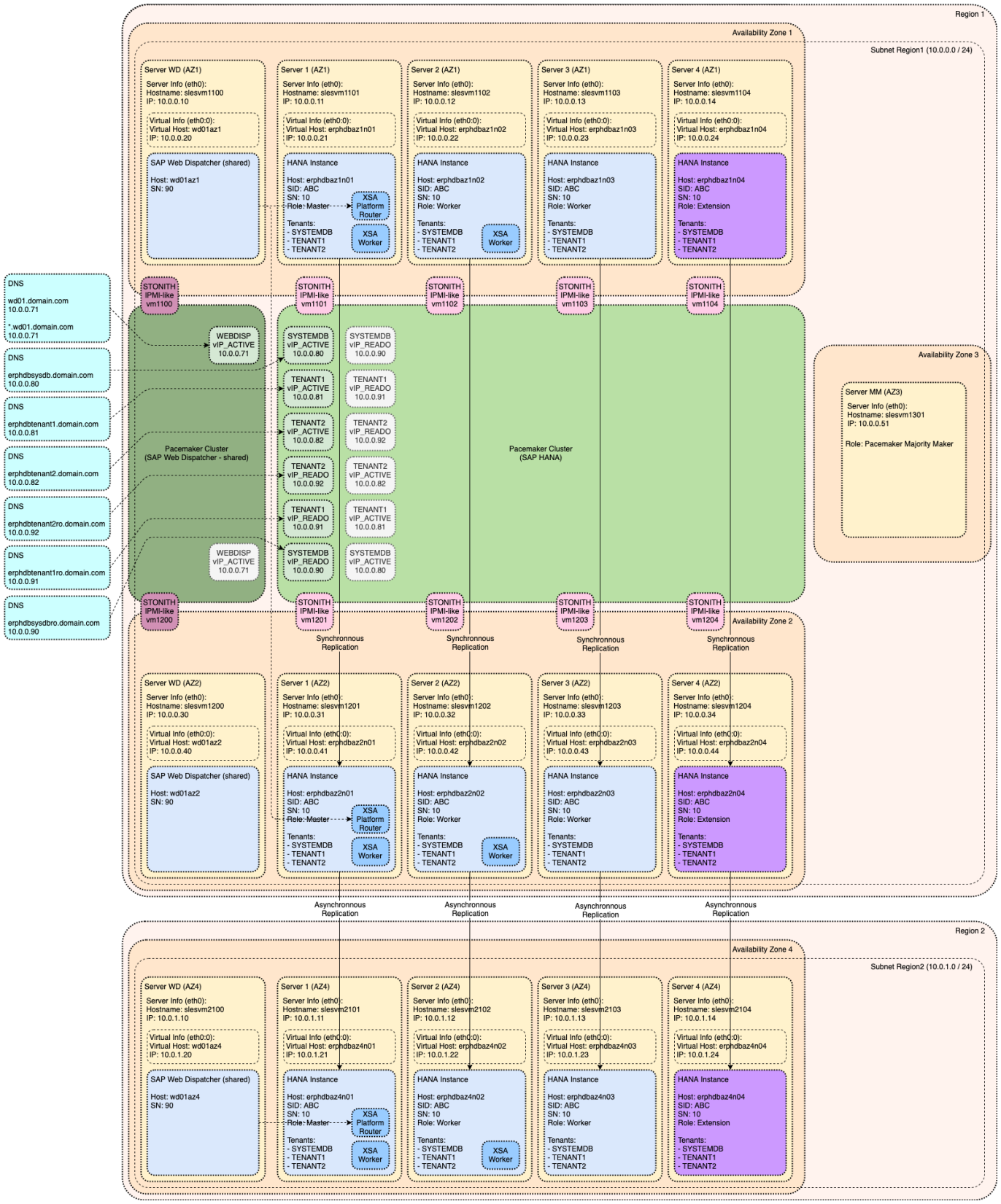


AWS: Implementation of Cluster IP

Platform Specific Architecture for Azure (Microsoft Azure)

Description

Azure: Overall Architecture



Azure: Implementation of Cluster IP