Desired state for complex applications

Jochen Kellner <jochen@jochen.org>
Nuremberg, 2025-06-26

What we’ll look at...

- Our vision
- VM/Server deployment: a solved problem
- Simple services like Apache/Tomcat or databases
- What we plan: describe an SAP System – simple and clustered
- At the horizon: SAP Landscapes
- Beyond the horizon: complex SAP Landscapes and their connections

Our vision for service configuration

- Describe a complete system with minimal input
- Generate a complete blueprint with conventions, standards, and derived options
- Deploy and manage the system with automation

VM/Server deployment: a solved problem

- VM/Server deployment: a solved problem
- IaaS: use terraform or opentofu
- Configure with cloud-init and automation
- What requirements have applications?

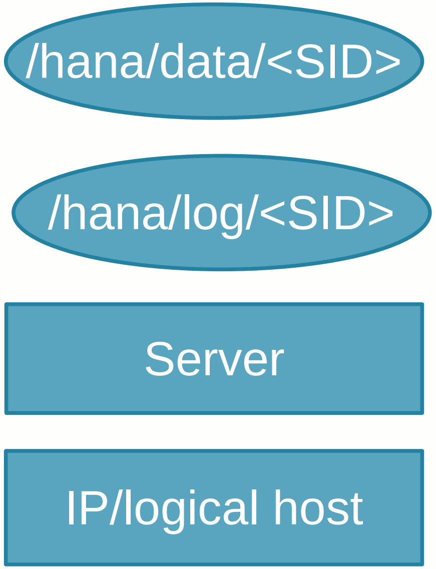
Simple services like Apache or databases

- What parameters describe the database (e.g. DB name)?
- Internal options due to standards, derived from input
- Why is that useful?
  - SAN names in certificates
  - Monitoring
  - Backup configuration, etc

Example: HANA database

- Input:
  - Database instance name <SID>
  - Database tenants
- Output:
  - OS user <sid>adm, tn_<tenant>adm
  - Filesystemes /hana/data/<SID>, /hana/log/<SID>,...
  - Logical host db<sid>, DNS, IP, SAN name in TLS certificate
  - Backup and monitoring configuration

Example: HANA database



/hana/data/<SID>

/hana/log/<SID>

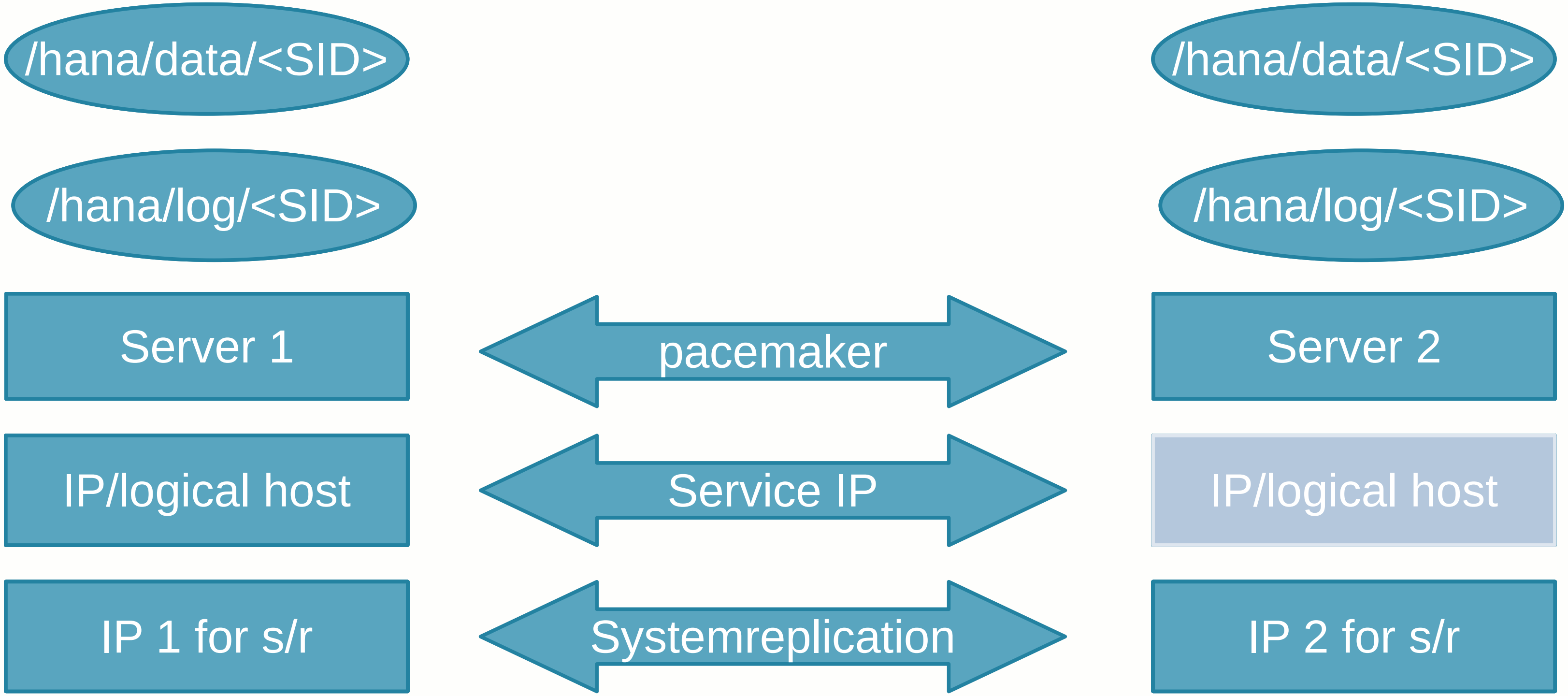
Server

IP/logical host

Example: HANA database cluster

- Additional input:
  - Database with system replication / cluster
- Derived additional output:
  - Two servers instead of one, second network for s/r
  - STONITH block devices, cluster software, cluster configuration
  - Some changes for network/DNS configuration
- Take away: to order a cluster you don’t need to specify details

Example: HANA database cluster



/hana/data/<SID>

/hana/log/<SID>

Server 1

IP/logical host

IP 1 for s/r

pacemaker

Service IP

Systemreplication

/hana/data/<SID>

/hana/log/<SID>

Server 2

IP/logical host

IP 2 for s/r

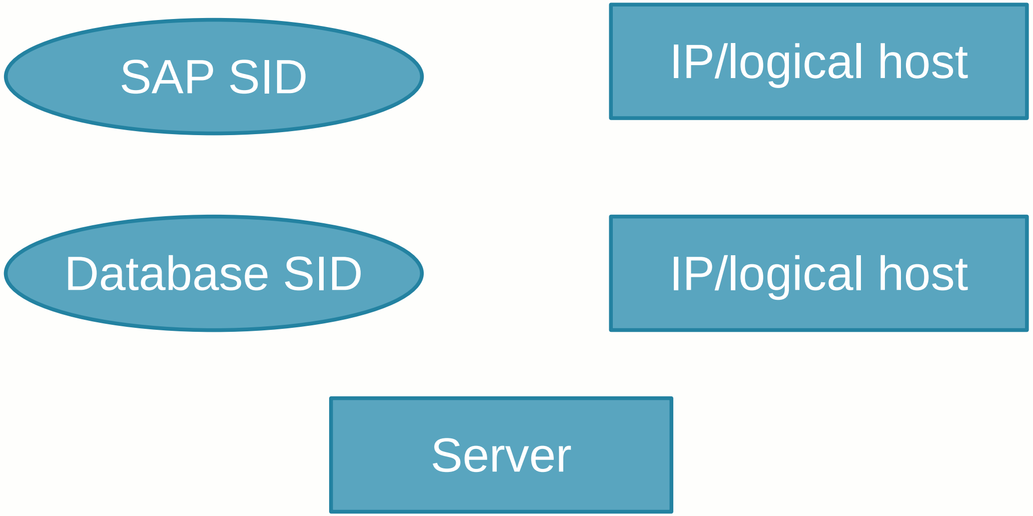
Tooling...

- Minimal input: data types, values, validation
- Each service (building block) has conventions and standards
- Derive the Bill of Materials, terraform, etc.
- Documentation for missing automation
  - RfC with standard input
  - Document manual steps
- For now: python scripts
- Automation with ansible/SALT/others

Describe an SAP System – simple

- Input:
  - SAP system id (SID)
  - Part of which landscape (e.g. software release, components)
  - Usage (e.g. production ⇒ disk mirror, backup schedule)
  - Database source: DVD or system copy
- Output:
  - logical hosts, IP addresses, DNS
  - Users
  - storage, filesystems, etc.

Example: SAP System



SAP SID

IP/logical host

Database SID

IP/logical host

Server

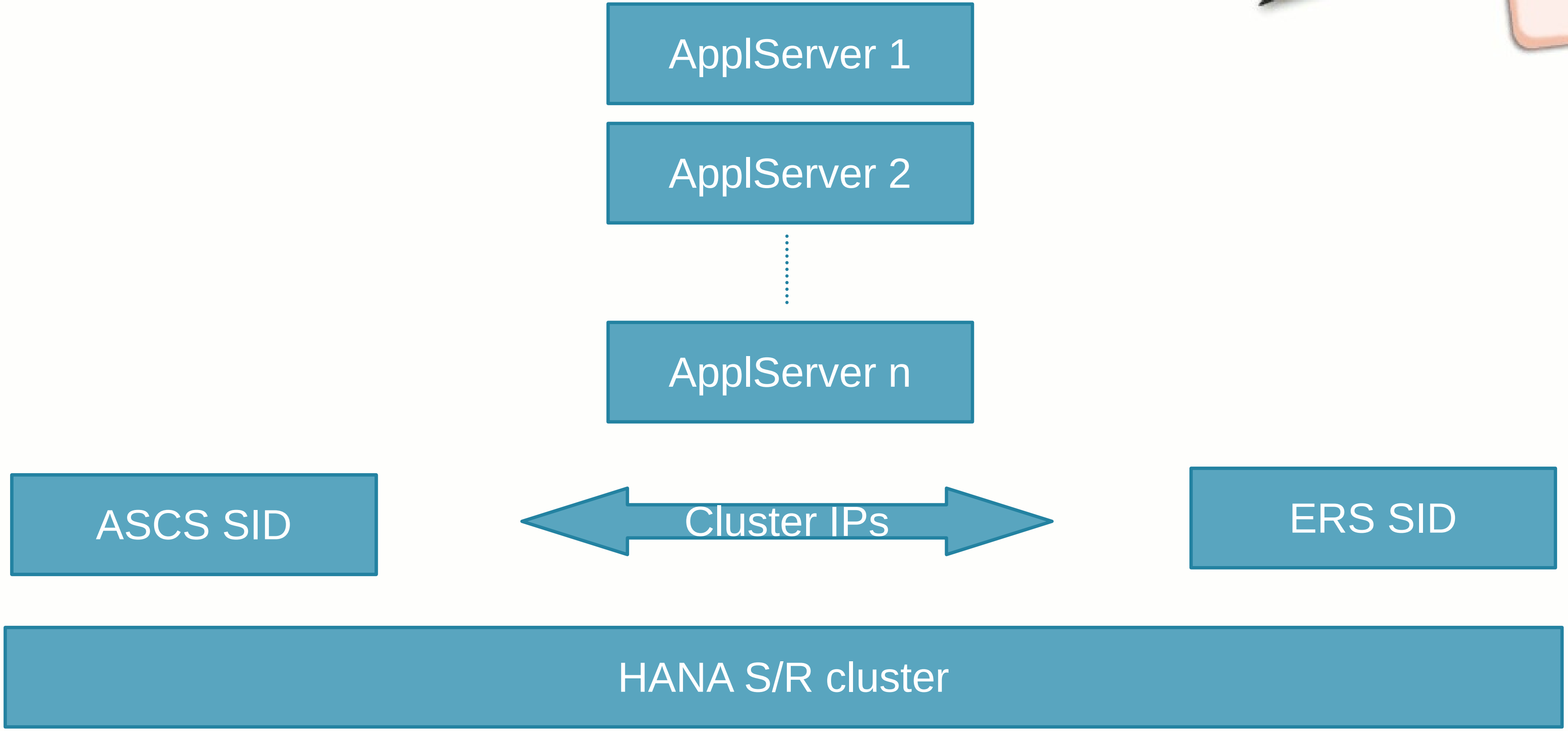
Describe an SAP System – templates

- Pre-defined architecture templates:
  - All in one – one VM with DB and SAP System
  - DB and SAP on different servers
  - Fully clustered SAP system and database
- Doing something different means:
  - Manual work
  - More potential for errors
  - Generates snowflakes

Describe an SAP System – clustered

- Additional Input:
  - Number of Appl Servers, CPU and RAM
- Output:
  - 2 VMs HANA S/R, 2 VMs for ASCS/ERS, n VMs as Appl Servers
  - Lots of logical hosts, IP addresses, DNS entries
  - Predefined storage/file systems
  - OS users

Example: SAP System



ApplServer 1

ApplServer 2

ApplServer n

ASCS SID

Cluster IPs

ERS SID

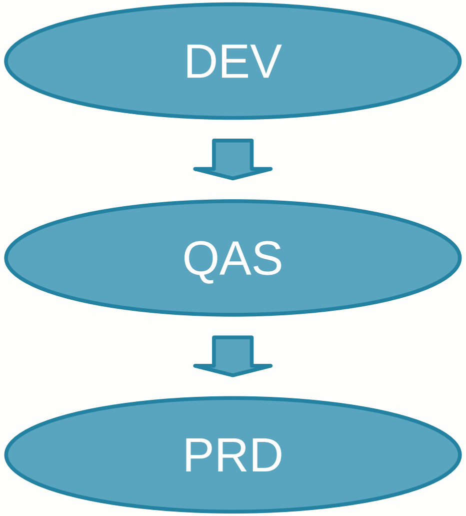
HANA S/R cluster

Describe an SAP System

- Until now: manual preparation of the building blocks, e.g. servers with database, ASCS, ERS, and application
- Future: Build me a complex system SID, generate the building blocks
- Why is that useful?
  - We got more and more systems of that complexity
  - We'll get even more clustered systems for higher availability
  - Manual building of the configuration is hard/error prone
  - Our automation makes deployment pretty easy and fast

At the horizon: SAP Landscapes

- What is an SAP system landscape?



- Development of custom code and customizing
- Export transport order to /usr/sap/trans
- Import into QA
- Test and release in QA
- Import into PRD, education,...

At the horizon: SAP Landscapes

- What describes an SAP system landscape?
- „Shared“ /usr/sap/trans, usually with NFS/SMB
- Transport domain controller, transport groups
- Same SAP software release, similar customizing and custom code, staged updates
- Internal communication (RFC, DEV ⇔ QAS ⇔ PRD)
- Similar communication destinations?
- VLANs

Beyond the horizon: Complex SAP Landscapes

- Multiple transport landscapes (loose/tight coupling?)
- Diverse SAP software releases
- May need consistent recovery, system copies, lockstep upgrades…
- Communication patterns/firewall rules:
  - Dev ERP ⇒ Dev BW
  - QA ERP ⇒ QA BW
  - Prod ERP ⇒ Prod BW

Conclusion

- Building blocks are useful to build more complex systems
- Minimum input makes systems easy to plan/order
- Defaults, conventions, and derived data are prerequisits to enable simple/minimal input and completly automated deployment/configuration
- Automatic deployment delivers speed and quality (less errors and better uniformity), but needs ongoing maintainance
- Complete description of a complex solution is possible, but a lot of work to achieve