SaaS Provisioning
Bau eines effizienten Systems

Sentinel™ Cloud Services

Michael MiZu Zunke

JFS 2010

Agenda

- Wer sind wir?
- Um was geht’s?
- Cloud – Was ist so anders? – Designkriterien
- Das Framework
- Transformation
  - Ein Beispiel
- Lessons learnt

SafeNet Fact Sheet

The largest company exclusively focused
on the protection of high-value information assets.
Protecting information from its creation throughout its lifecycle.

- **Founded:** 1983
- **Ownership:** private
- **Global Footprint** with more than 25,000 customers in 100 countries
- **Employees:** 1550 in 25 countries
- **Accredited** with products certified to the highest security standards

Sentinel®

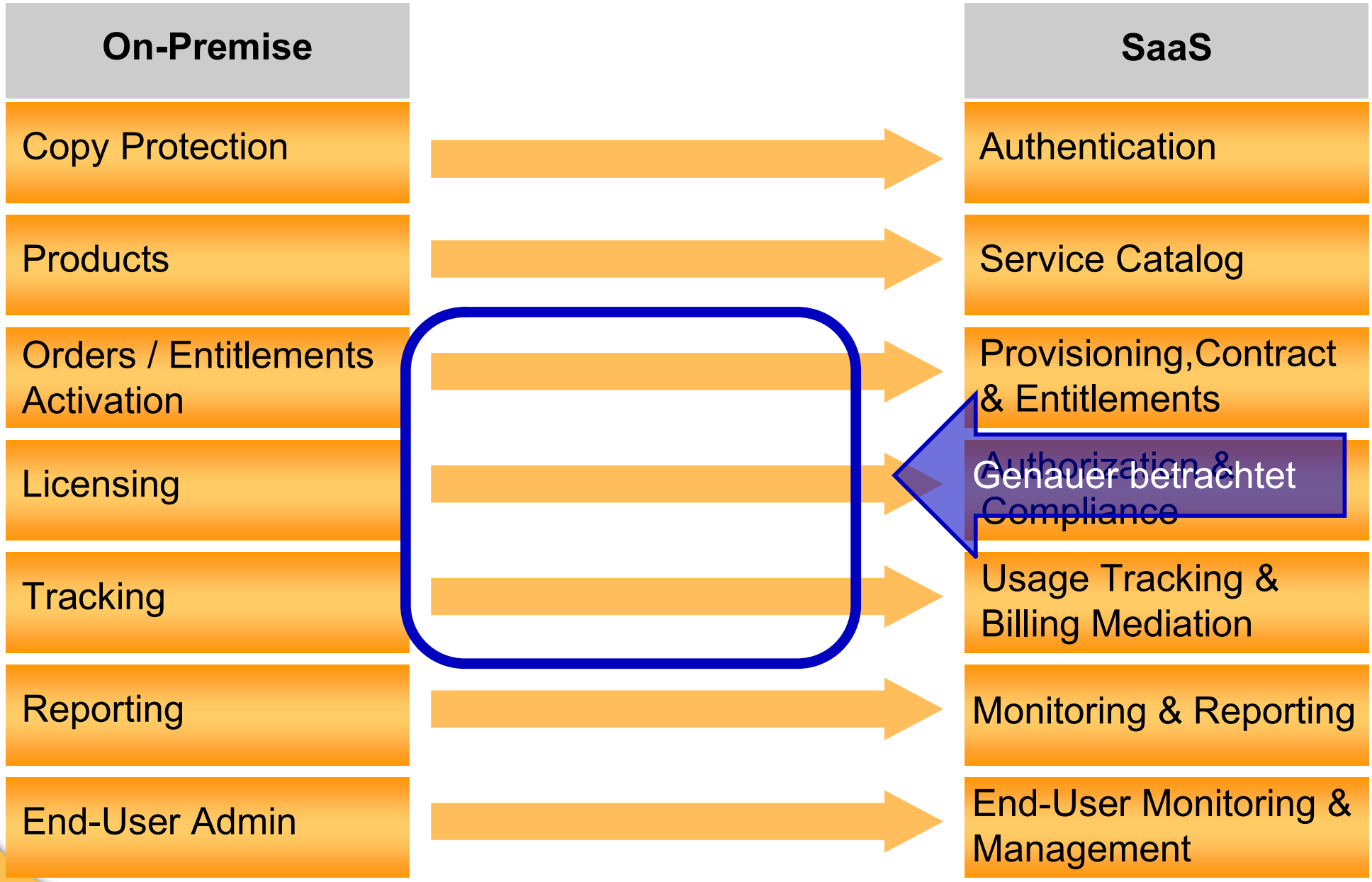
Sentinel HASP™

Flexible Solutions for Evolving Business

for your **Business** | for your **Customers** | for your **Budget** | for your **Future**

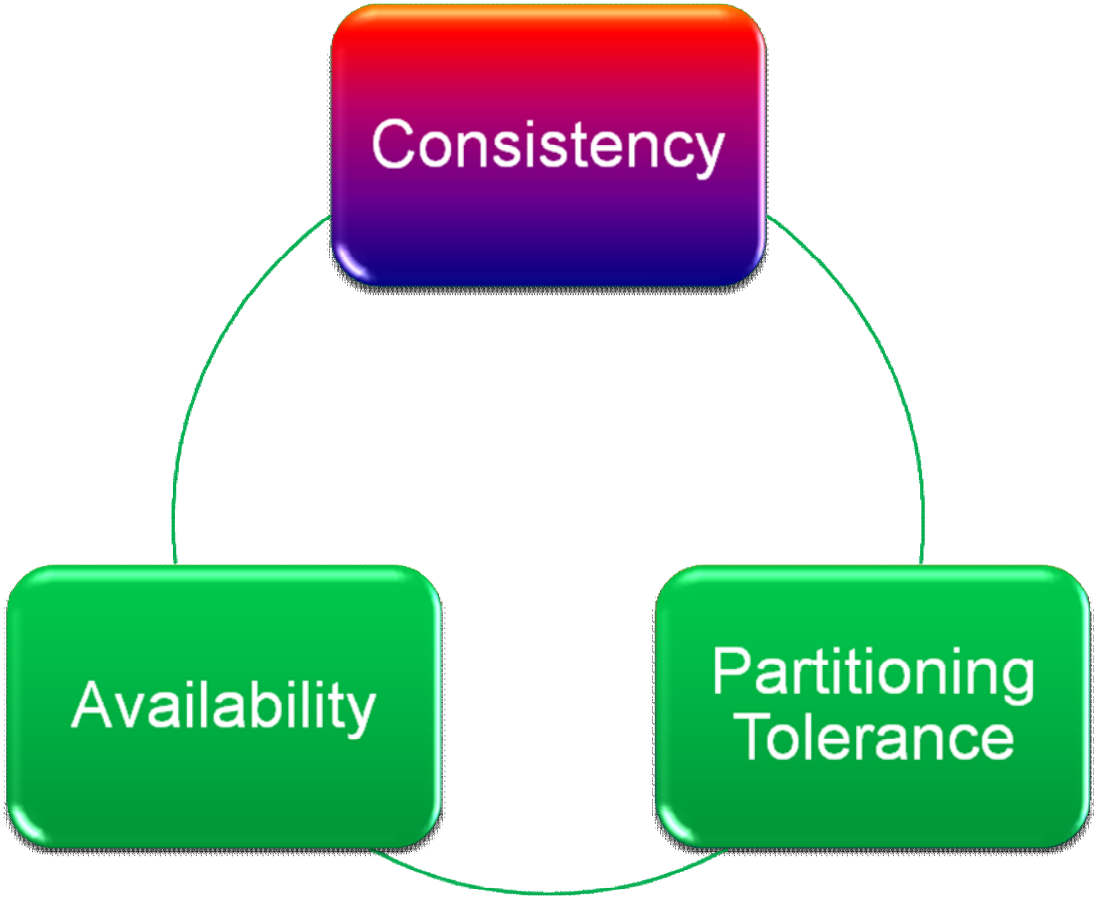
- ✓ Embrace emerging business models to **reach valuable new markets**
- ✓ Protect against reverse engineering to **maintain competitive advantage**
- ✓ Prevent unauthorized use and distribution to **safeguard existing revenue**
- ✓ Automate operational tasks to **save time**, **minimize errors**, **and reduce costs**
- ✓ Simplify reporting and compliance to **improve partner** and **end-user relations**

Parallelen


On-Premise
SaaS
Copy Protection
Authentication
Products
Service Catalog
Orders / Entitlements Activation
Provisioning,Contract & Entitlements
Licensing
Authorization & Compliance
Genauer betrachtet
Tracking
Usage Tracking & Billing Mediation
Reporting
Monitoring & Reporting
End-User Admin
End-User Monitoring & Management

Zuerst ein paar Grundlagen

Das CAP Theorem



CAP – was tun?

- Skalierbarkeit durch niedrige Priorität auf Konsistenz
- „Eventually consistent“ Ansatz
- Siehe auch:
  http://www.allthingsdistributed.com/2008/12/eventually_consistent.html

Folgerungen für unsere Problem Domain

- Manche klassischen Lizenzmodell skalieren nicht
  - Floating Lizenz, Deadcounter
  - Erfordern eine *global limitierte* Resource
  - Erfordern also globale Synchronisation!

- Wie macht man sie Cloud-Kompatibel?

Paradigmen unserer SaaS Architektur

- **No Delay!**

  speed, bandwidth & reaction time of service must not suffer just because the service is licensed.

Paradigmen unserer SaaS Architektur

- **Scale with the customer!**

  No additional infrastructure because of licensing system.

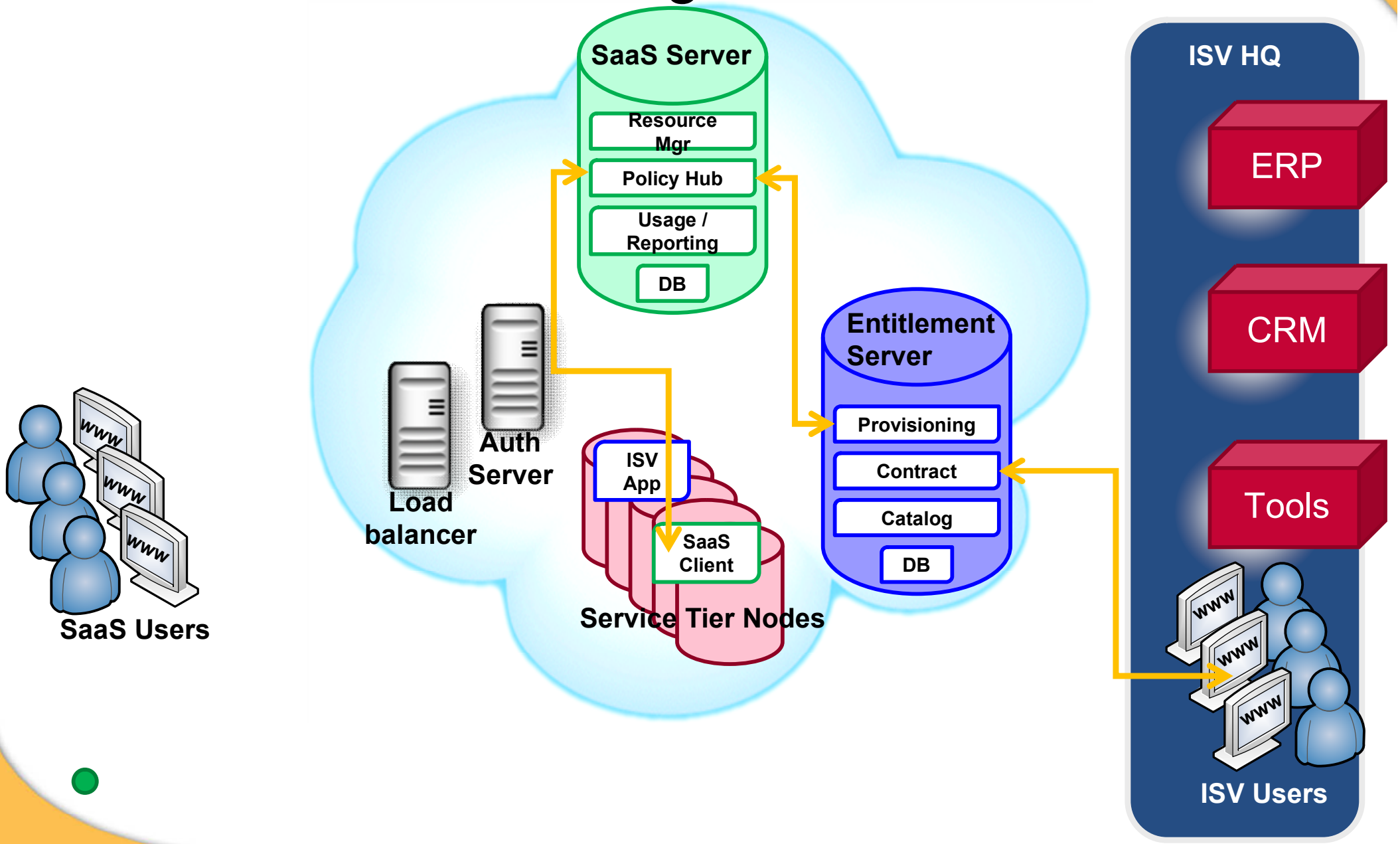
Paradigmen unserer SaaS Architektur

- **No Network Assumptions!**
  - **Don't assume, the network connection is a given. The system must work without network.**
  - **Temporary disconnects should be handled within the fault tolerance physics!**

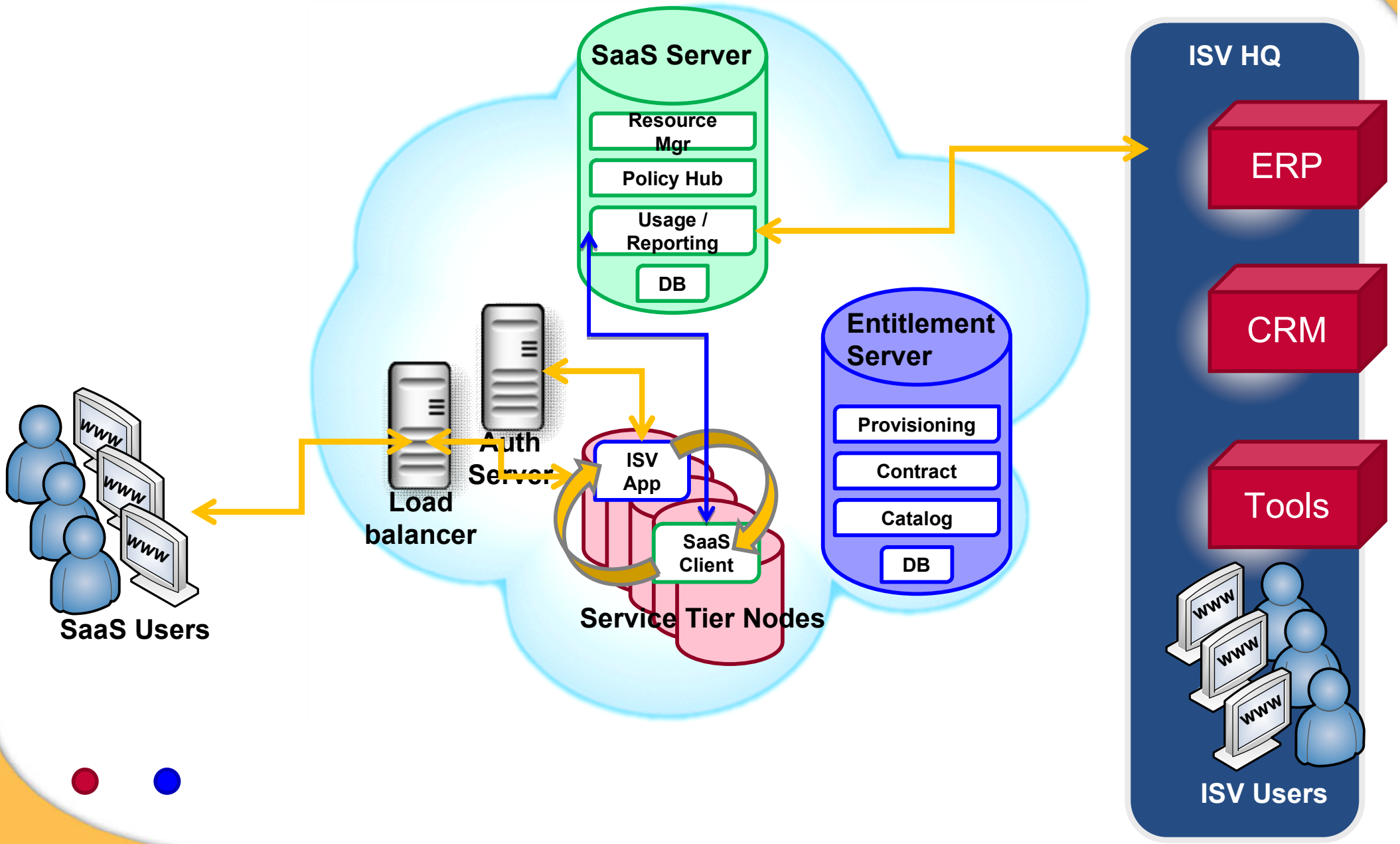
Paradigmen unserer SaaS Architektur

- **No sharp Cut-Off in distributed license resource data!**
  - **Because of the "No Delay!" paradigm the system cannot guarantee sharp cut-offs in absolute globally shared values like executions and concurrency values.**
  - **If enterprises want sharp boundaries, they have to tolerate a drawback on service quality.**

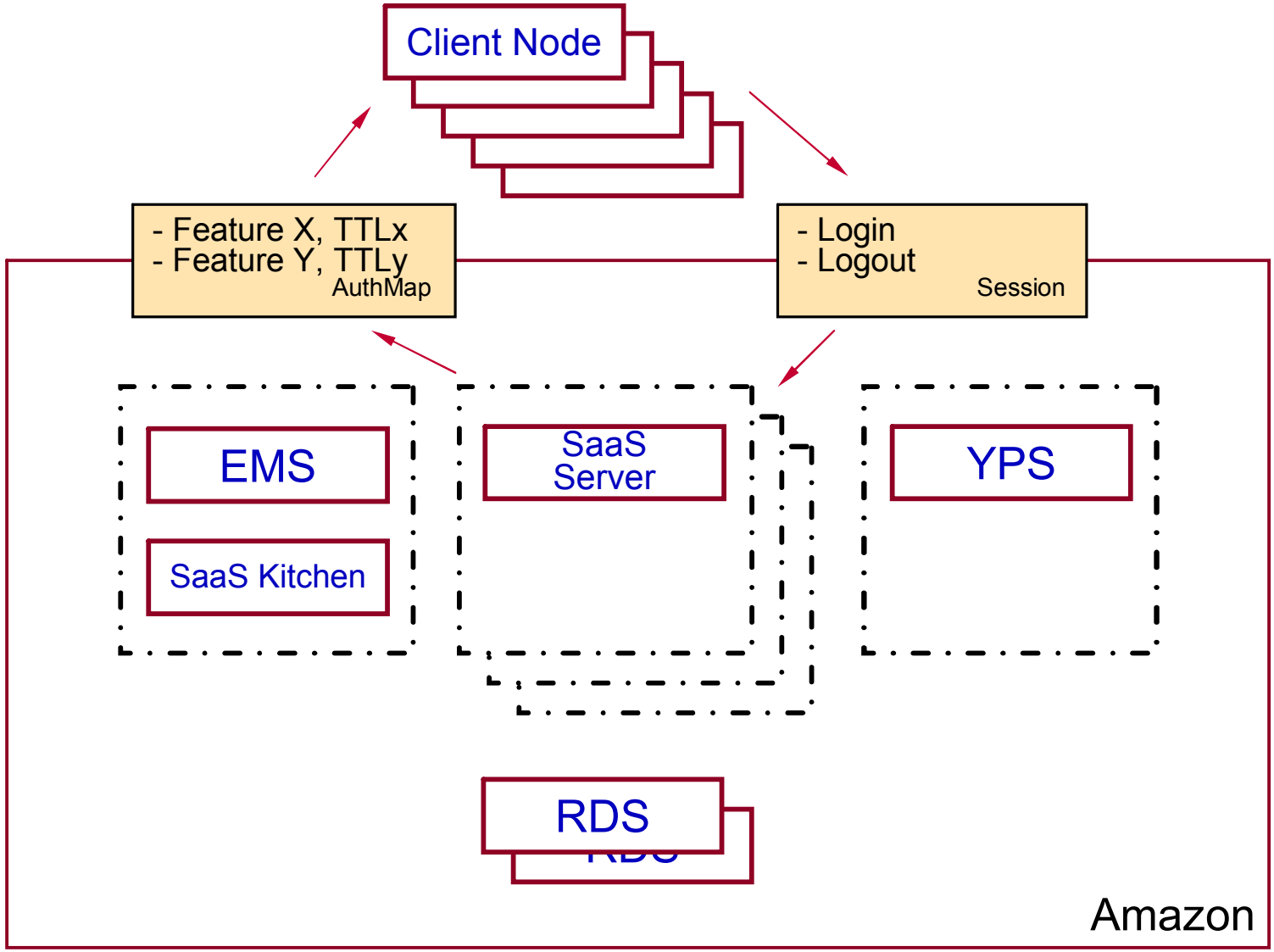
Ablauf Provisionierung



Ablauf Metered Use



SaaS „Sentinel Cloud“ im Detail



Lesons learnt

- Lokale Intelligenz reduziert externe Abhängigkeit
- Stateless eliminiert Synchronisierung
- Caching & asynchrones update entkoppelt Module

- Manche klassischen Modelle müssen an die Cloud angepaßt werden um „*eventually consistent*“ zu werden!

Fragen?