OpenStack – Automatisiertes Bereitstellen von Instanzen

Linux Informationstag Augsburg 2014

22. März 2014

Ralph Dehner
Gründer & CEO
B1 Systems GmbH
dehner@b1-systems.de

Inhalt

- Vorstellung
- Die Cloud und ihre Eigenschaften
- OpenStack
- Orchestrierung

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 60 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Entwicklung
  - Training
  - Betrieb
  - Lösungen
- dezentrale Strukturen

Schwerpunkte

- Virtualisierung (XEN, KVM & RHEV)
- Systemmanagement (Spacewalk, Red Hat Satellite, SUSE Manager)
- Konfigurationsmanagement (Puppet & Chef)
- Monitoring (Nagios & Icinga)
- IaaS Cloud (OpenStack & SUSE Cloud)
- Hochverfügbarkeit (Pacemaker)
- Shared Storage (GPFS, OCFS2, DRBD & CEPH)
- Dateiaustausch (ownCloud)
- Paketierung (Open Build Service)
- Administratoren oder Entwickler zur Unterstützung des Teams vor Ort

Die Cloud und ihre Eigenschaften

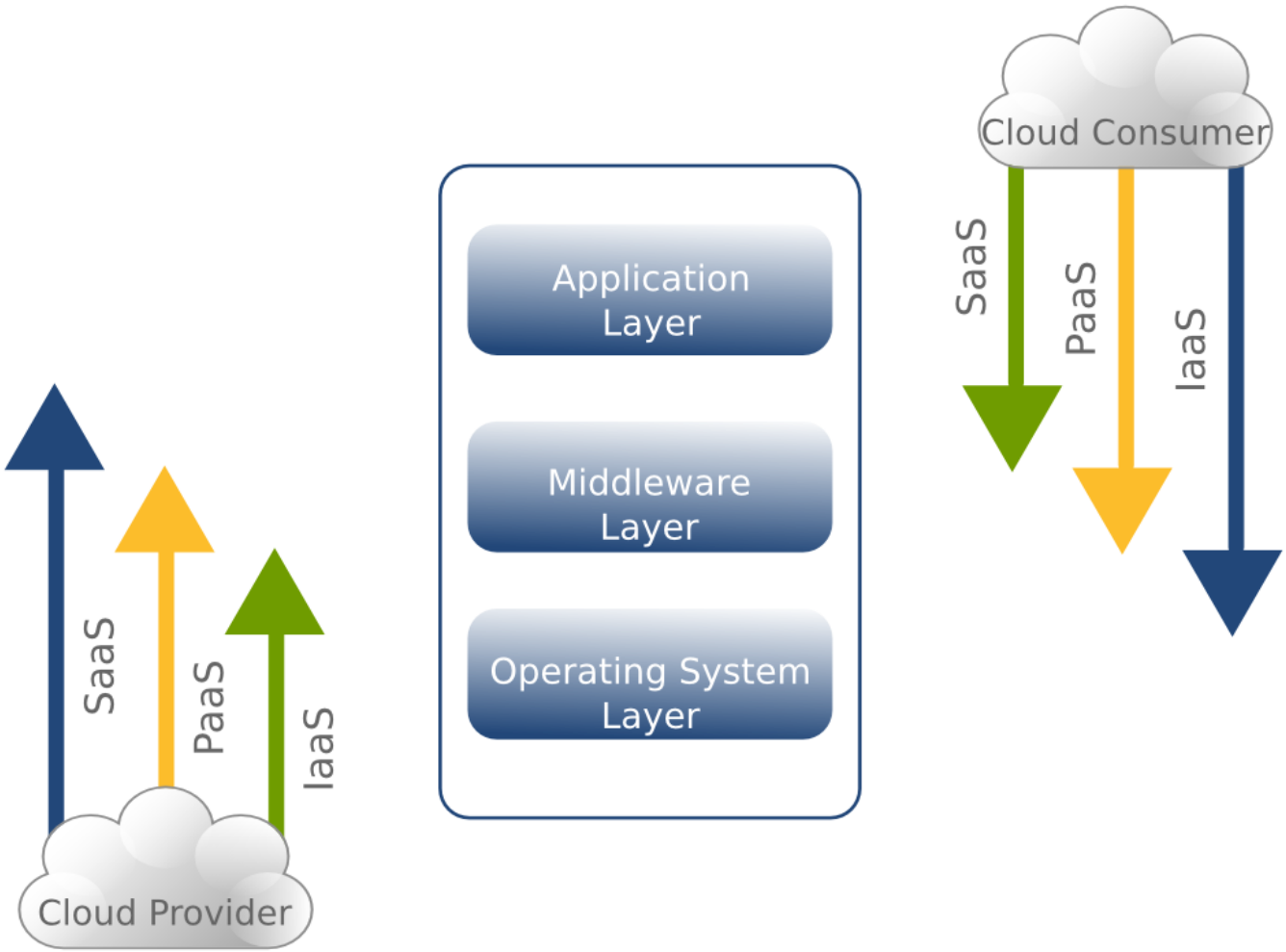
Eigenschaften nach NIST

- On-Demand Self Service
- Broad Network Access
- Resource Pooling
- Rapid Elasticity
- Measured Services

Bereitstellungs-/Liefermodelle

- Private Cloud
- Public Cloud
- Community Cloud
- Hybrid Cloud

Servicemodelle



Definition nach BSI

Cloud Computing bezeichnet das dynamisch an den Bedarf angepasste Anbieten, Nutzen und Abrechnen von IT-Dienstleistungen über ein Netz. Angebot und Nutzung dieser Dienstleistungen erfolgen dabei ausschließlich über definierte technische Schnittstellen und Protokolle. Die Spannbreite der im Rahmen von Cloud Computing angebotenen Dienstleistungen umfasst das komplette Spektrum der Informationstechnik und beinhaltet unter anderem Infrastruktur (z.B. Rechenleistung, Speicherplatz), Plattformen und Software.

OpenStack

Merkmale

- Infrastructure as a Service (IaaS)
- Open Source Software
- mehrere Kernprojekte (Virtualisierung, Netzwerk, Storage, . . .)
- mandantenfähig
- Unterstützung durch viele Firmen (IBM, HP, Rackspace, Cisco, . . .)
- gegründet im Juli 2010
- B1 Systems ist Upstream Contributor seit Anfang 2011
- aktuelles Release *Havana* (2013.2)
- kommendes Release *Icehouse* (2014.1) voraussichtlich April 2014

Komponenten

Compute (Nova) verwaltet virtuelle Maschinen

Object Storage (Swift) stellt Object Storage zur Verfügung

Block Storage (Cinder) gewährt Instanzen (VMs) Block Storage

Networking (Neutron) managt Netzwerke und entsprechende Komponenten

Dashboard (Horizon) Webinterface

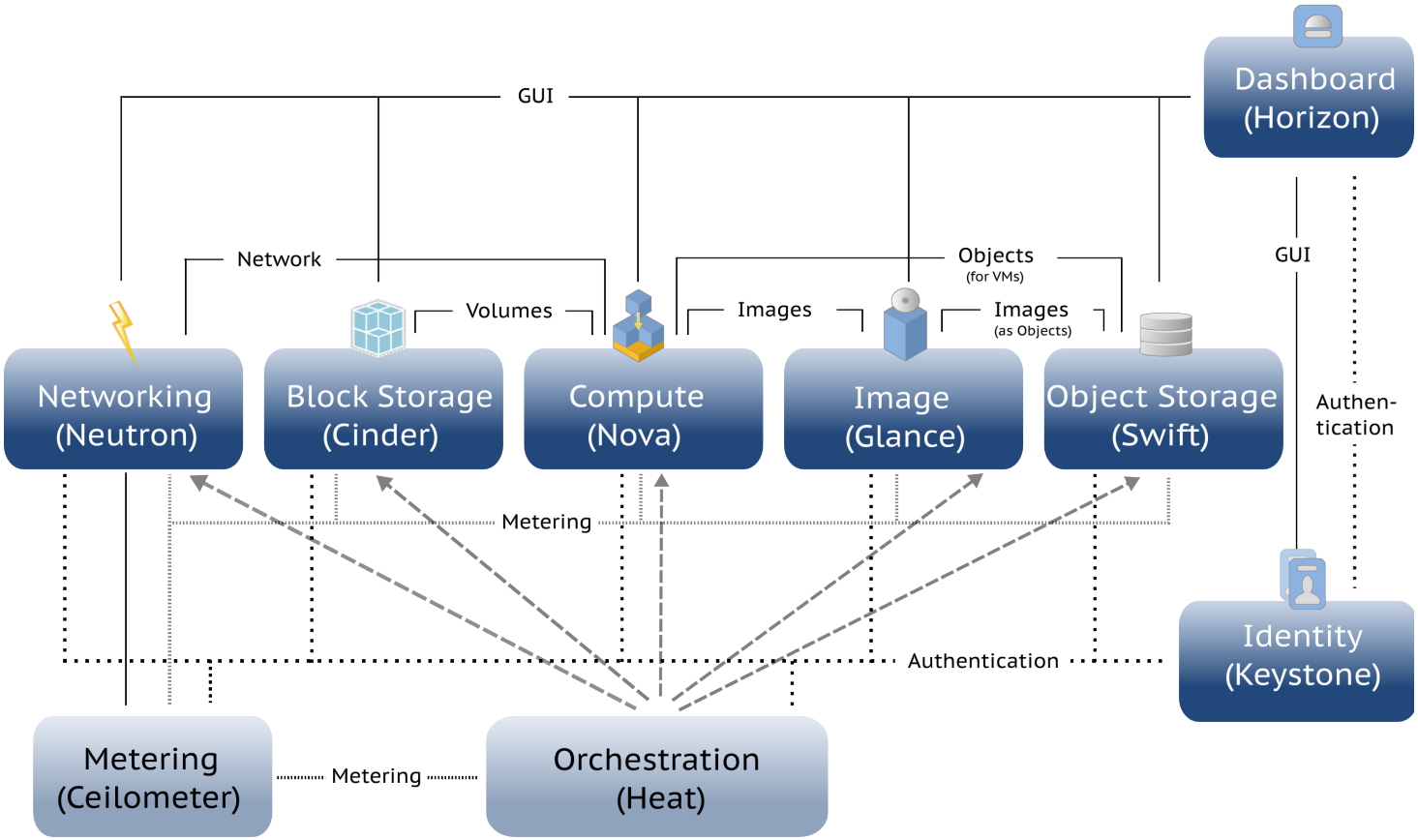
Identity (Keystone) Benutzerverwaltung

Image (Glance) verwaltet Images für virtuelle Maschinen

Telemetry (Ceilometer) erfasst Nutzungs- und Performancedaten

Orchestration (Heat) Template-basierte Orchestrierung

Komponentenübersicht



Deployment

Vereinfachtes Deployment

1. Anfrage an Nova-API
2. Scheduler sucht passenden Host
3. Netzwerkkonfiguration wird gesetzt (Netzwerkinfrastruktur)
4. Kopieren des Image auf den Hypervisor
5. Start der VM
6. Anbindung von Storage
7. Config-Management (in der VM)

Deployment ganzer Infrastrukturen

- Umgebungen bestehen aus mehreren virtuellen Maschinen, die unterschiedliche Eigenschaften besitzen (vCPUs, RAM, Netzwerkkarten, . . .)
- zusätzliche Netzwerkkonfigurationen müssen vorgenommen werden (VPN, Routing)
- eventuell Einbindung von zusätzlichem Block Storage
- Konfiguration der Software in den VMs
- Orchestrierung als mögliche Lösung

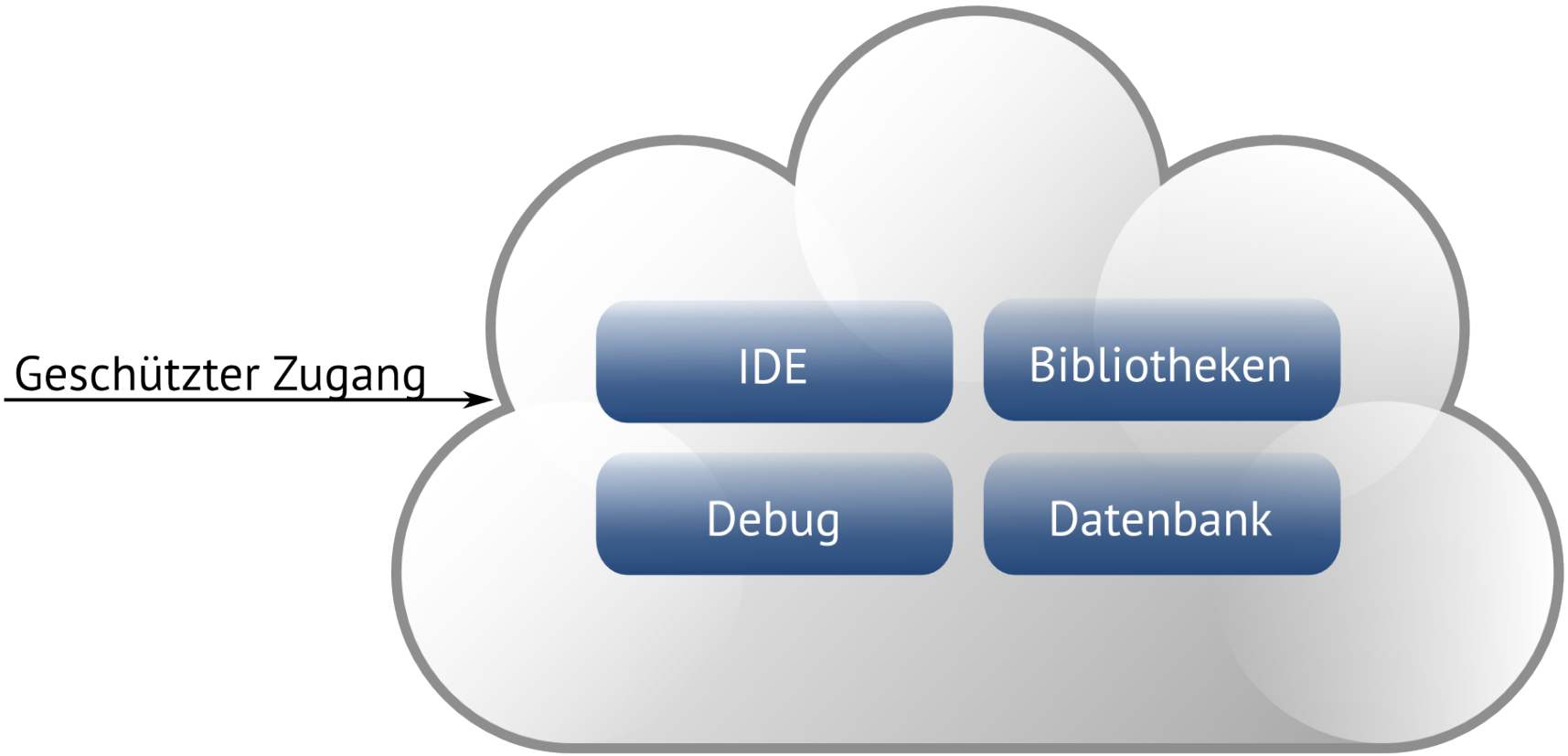
Beispielumgebungen

Entwicklungsumgebungen

Entwicklungsumgebungen haben meist folgende Eigenschaften:

- oftmals nur intern erreichbar (gewollter und geschützter Zugriff)
- benötigen zusätzliche Software (Bibliotheken, Entwicklungsumgebungen, Versionsverwaltung etc.)
- abhängige Systeme oft nur rudimentär bereitgestellt (Datenbanken nicht optimiert)
- Analyse der entwickelten Anwendungen im Debug-Modus
- werden schnell erweitert

Entwicklungsumgebungen



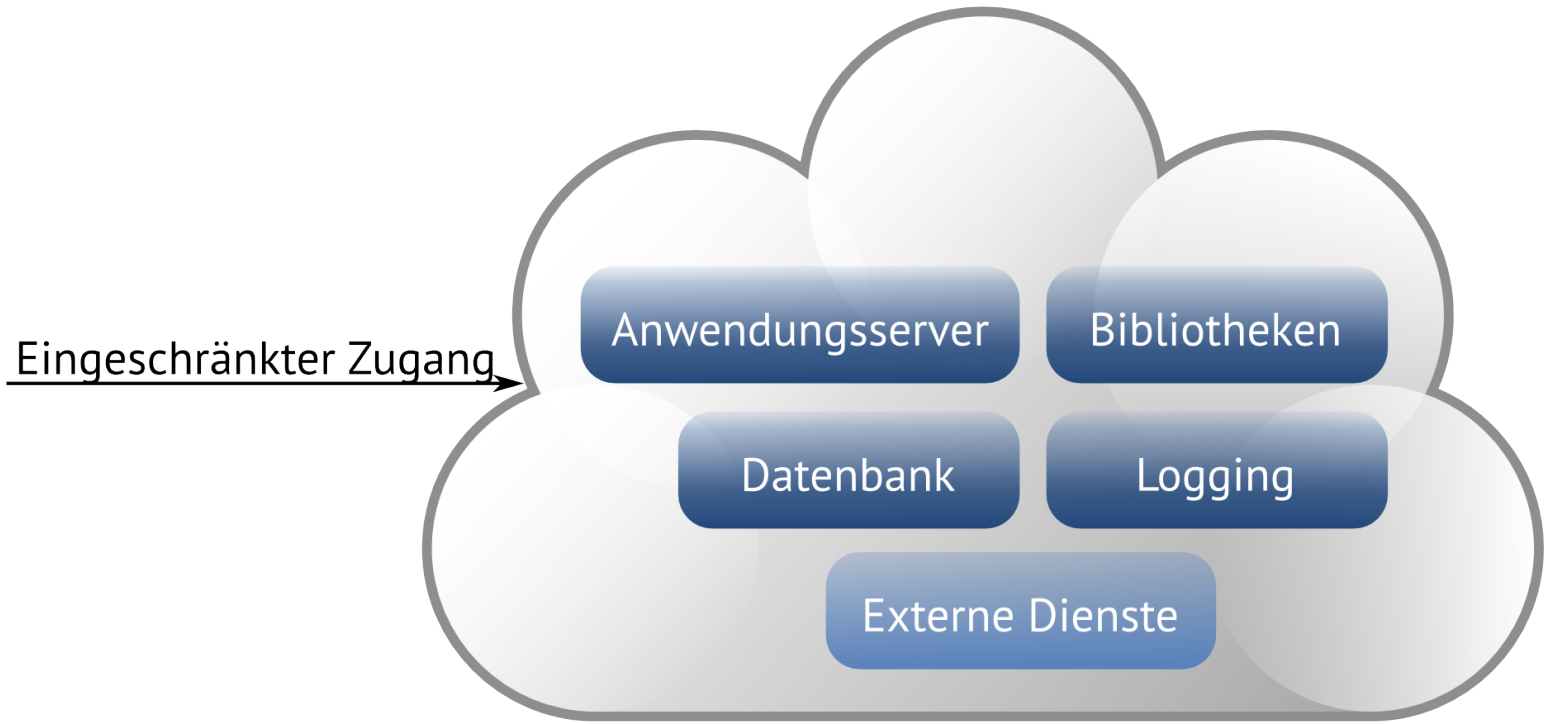
Protokollierung der Anwendung im Debug-Modus

Testumgebungen

Testumgebungen näheren sich den Eigenschaften der Produktivumgebungen an, zusätzlich sollte ein breiter Benutzerkreis Software testen:

- benötigen zusätzliche Software (Profiling von Zugriffen, Debuginformationen etc.)
- eventuell Anbindung an externe Dienste (Authentifizierung, Datenbanken)
- Abbild der Produktivumgebung (optimal)
- Protokollierung im Debug-Modus
- Anwendungstester sind keine Programmierer

Testumgebungen

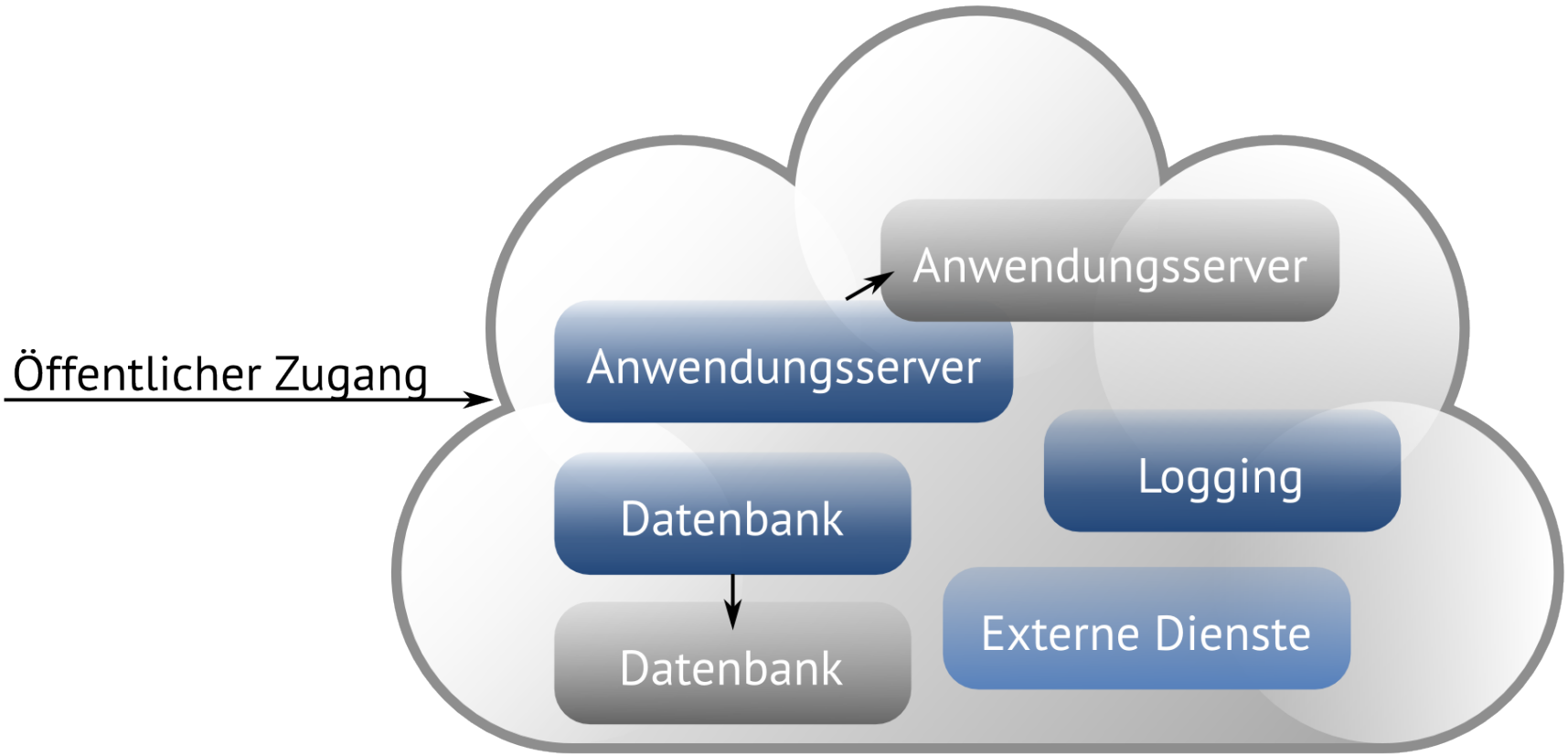


Produktivumgebungen

Produktivumgebungen haben strengere Eigenschaften und weitere Möglichkeiten, sie bieten nur die lauffähigen Komponenten an:

- keine Entwicklungstools (Sicherheit, Paketmanagement, Updates)
- Anbindung externer Dienste
- vielfach öffentlich erreichbar
- Monitoring der Last
- skalierend

Produktivumgebungen



Orchestrierung

Was bedeutet das?

- Automatisierung von Aufgaben, die Ressourcen managen, in einem Workflow
- betrifft die Interaktion mit Infrastrukturkomponenten (per API)
- sorgt für die Konfiguration der Ressourcen (Bereitstellung von Storage, . . .)
- ermöglicht die Konfiguration von Software innerhalb von VMs
- aktualisiert nach Möglichkeit Ressourcen, ohne alle laufenden zu verändern

Beispiel ohne Orchestrierung

Schritte für das Bereitstellen einer Testumgebung:

1. Erzeugen und Starten Anwendungsserver
2. Erzeugen und Starten Datenbankserver
3. Konfiguration Netzwerk
4. Konfiguration Logging
5. Routing zu externen Diensten konfigurieren
6. Bereitstellung VPN

Das Löschen erfordert erneut alle Schritte.

Beispiel mit Orchestrierung

1. Erzeugen eines wiederverwendbaren Template
2. mit Orchestrierung ein Aufruf mit dem Template
3. Löschen aller Ressourcen in einem Aufruf per Template

Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Eigenschaften

- Orchestrierung mit Templates
- Templates können die gesamte Infrastruktur beschreiben (Instanzen, Netzwerk, Storage, Monitoring, . . .)
- durch Nutzung parametrisierter Templates können diese beim Aufruf angepasst werden
- Einbinden externer Konfigurationen ist hier möglich
- Templates können also vollständig Entwicklungs-, Test-, und Produktionsumgebungen beschreiben und starten/stoppen
- Lifecycle-Management einer Anwendung durch Heat

Begriffe

Template Textdatei, die einen Stack beschreibt (HOT (*Heat Orchestration Template*) oder CFN (AWS *Cloud Formation*))

Stack Summe aller Ressourcen – aus einem Template – die notwendig sind, um eine Umgebung aufzubauen

Resource alle Ressourcen innerhalb der OpenStack-Umgebung – IP-Adressen, Ports, VMs, Images, Volumes etc.

Event einzelne Aufgabe einer Ressource – Erzeugen eines Volume, Attach, Detach, Löschen

Funktionsweise

1. Starten eines Stack per übergebenem Template (Benutzer adressiert die Heat-API)
2. Auflösen der Abhängigkeiten im Template
3. Adressieren der einzelnen OpenStack-APIs (Network, Storage, . . .) um die einzelnen Ressourcen zu erstellen

Interessante Features

- Suspend/Resume von Stacks ist möglich – dadurch können unterschiedliche Umgebungen vorgehalten werden
- Update von vielen Ressourcen machbar – kein Löschen, sondern Modifizieren der laufenden Ressource
- Durch Konfiguration von Alarmen (Ceilometer) kann automatische Skalierung erreicht werden

Interessante Features

- Suspend/Resume von Stacks ist möglich – dadurch können unterschiedliche Umgebungen vorgehalten werden
- Update von vielen Ressourcen machbar – kein Löschen, sondern Modifizieren der laufenden Ressource
- Durch Konfiguration von Alarmen (Ceilometer) kann automatische Skalierung erreicht werden

Interessante Features

- Suspend/Resume von Stacks ist möglich – dadurch können unterschiedliche Umgebungen vorgehalten werden
- Update von vielen Ressourcen machbar – kein Löschen, sondern Modifizieren der laufenden Ressource
- Durch Konfiguration von Alarmen (Ceilometer) kann automatische Skalierung erreicht werden

Template-Beispiel

```
[...]
  Instance_Database:
    type: OS::Nova::Server
    properties:
      image: SUSE Linux Enterprise 11
      flavor: m1.tiny
      networks:
        - port: { get_resource: Instance_Port_Database }

  Instance_Port_Database:
    type: OS::Neutron::Port
    properties:
      network_id: { get_param: Network }
      fixed_ips:
        - subnet_id: { get_param: Subnet }
[...]
```

Stack-Zustand

```
+------------------------------+----------------------+----------+
| resource_name                | resource_type        | status   |
+------------------------------+----------------------+----------+
| FloatingIP_Webserver         | FloatingIP           | COMPLETE |
| Instance_Port_Database       | Port                 | COMPLETE |
| Instance_Port_Webserver      | Port                 | COMPLETE |
| FloatingIP_Associate_Webserver| FloatingIPAssociation| COMPLETE |
| Storage_Volume               | Volume               | COMPLETE |
| Instance_Database            | Server               | COMPLETE |
| Volume_Attachment            | VolumeAttachment     | COMPLETE |
| Instance_Webserver           | Server               | COMPLETE |
+------------------------------+----------------------+----------+
```

Vorgehensweise

1. Definieren Sie Ihre Umgebungen (VM-Größen, Images, Netzwerk, Storage, Software).
2. Planen Sie ein parametrisiertes Template für verschiedene Umgebungen.
3. Erzeugen Sie Heat-Templates anhand Ihrer Anforderungen (Skalierbarkeit, Alarme, Gruppen, . . .).
4. Starten Sie Ihre Umgebungen mit *einem* Stack.

Vorgehensweise

1. Definieren Sie Ihre Umgebungen (VM-Größen, Images, Netzwerk, Storage, Software).
2. Planen Sie ein parametrisiertes Template für verschiedene Umgebungen.
3. Erzeugen Sie Heat-Templates anhand Ihrer Anforderungen (Skalierbarkeit, Alarme, Gruppen, . . .).
4. Starten Sie Ihre Umgebungen mit *einem* Stack.

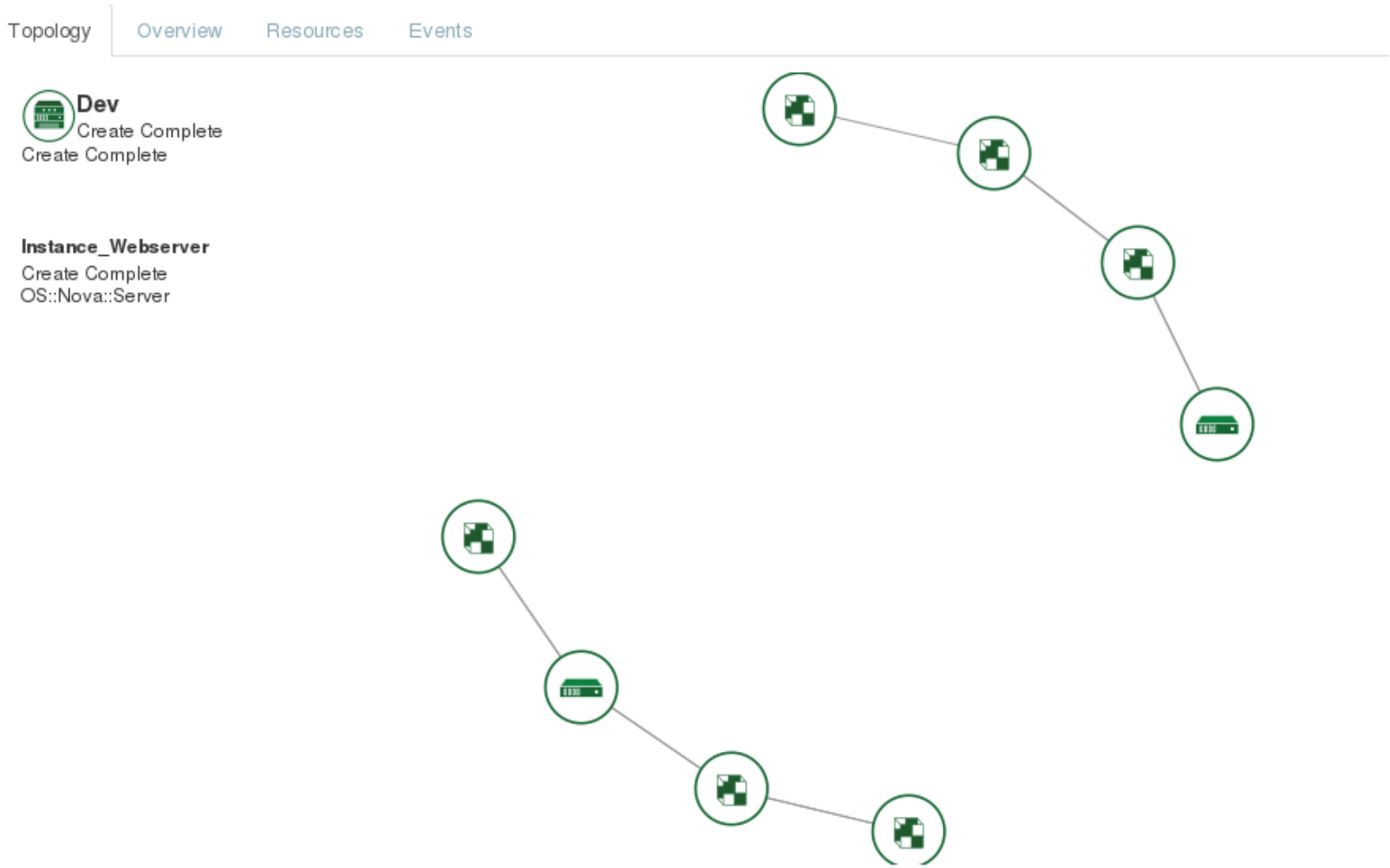
Vorgehensweise

1. Definieren Sie Ihre Umgebungen (VM-Größen, Images, Netzwerk, Storage, Software).
2. Planen Sie ein parametrisiertes Template für verschiedene Umgebungen.
3. Erzeugen Sie Heat-Templates anhand Ihrer Anforderungen (Skalierbarkeit, Alarme, Gruppen, . . .).
4. Starten Sie Ihre Umgebungen mit *einem* Stack.

Vorgehensweise

1. Definieren Sie Ihre Umgebungen (VM-Größen, Images, Netzwerk, Storage, Software).
2. Planen Sie ein parametrisiertes Template für verschiedene Umgebungen.
3. Erzeugen Sie Heat-Templates anhand Ihrer Anforderungen (Skalierbarkeit, Alarme, Gruppen, . . .).
4. Starten Sie Ihre Umgebungen mit *einem* Stack.

Stack Topologie



Stack Ressourcen

Stack Resources

Stack Resource	Resource	Stack Resource Type	Date Updated	Status	Status Reason
FloatingIP_Webserver	30105624-baff-4dfd-b9f9-3de4639b9ac0	OS::Neutron::FloatingIP	3 minutes	Create Complete	state changed
Storage_Volume	1cee95df-b4e6-4b48-92be-062743ca441d	OS::Cinder::Volume	3 minutes	Create Complete	state changed
Instance_Port_Webserver	e44b9213-3f79-43dc-982d-792b4a3d9777	OS::Neutron::Port	3 minutes	Create Complete	state changed
Instance_Webserver	ff84eb12-3d23-41ba-99ea-549f29f94525	OS::Nova::Server	1 minute	Create Complete	state changed
Instance_Port_Database	5a0c81f1-5305-4697-80a5-51c170381cb9	OS::Neutron::Port	3 minutes	Create Complete	state changed
Instance_Database	4fc0396c-78c3-42e6-9922-a25086257f37	OS::Nova::Server	1 minute	Create Complete	state changed
FloatingIP_Associate_Webserver	30105624-baff-4dfd-b9f9-3de4639b9ac0:e44b9213-3f79-43dc-982d-792b4a3d9777	OS::Neutron::FloatingIPAssociation	3 minutes	Create Complete	state changed
Volume_Attachment	1cee95df-b4e6-4b48-92be-062743ca441d	OS::Cinder::VolumeAttachment	1 minute	Create Complete	state changed

Stack Events

Stack Events

Stack Resource	Resource	Time Since Event	Status	Status Reason
Storage_Volume	-	2 minutes	Create In Progress	state changed
FloatingIP_Webserver	-	2 minutes	Create In Progress	state changed
Instance_Port_Database	-	2 minutes	Create In Progress	state changed
Instance_Port_Webserver	-	2 minutes	Create In Progress	state changed
FloatingIP_Webserver		2 minutes	Create Complete	state changed
Storage_Volume		2 minutes	Create Complete	state changed
Instance_Port_Database		2 minutes	Create Complete	state changed
Instance_Port_Webserver		2 minutes	Create Complete	state changed
FloatingIP_Associate_Webserver	-	2 minutes	Create In Progress	state changed
Instance_Database	-	2 minutes	Create In Progress	state changed
Instance_Webserver	-	2 minutes	Create In Progress	state changed
FloatingIP_Associate_Webserver		2 minutes	Create Complete	state changed
Instance_Database		0 minutes	Create Complete	state changed
Volume_Attachment	-	0 minutes	Create In Progress	state changed
Instance_Webserver		0 minutes	Create Complete	state changed
Volume_Attachment		0 minutes	Create Complete	state changed

Links

OpenStack `http://www.openstack.org/`

Heat-Wiki `https://wiki.openstack.org/wiki/Heat`

Heat-Template-Beispiele `https://github.com/openstack/heat-templates`

Template-Guide `http://docs.openstack.org/developer/heat/template_guide/`

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096