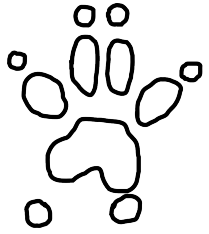
Entitlements

In a Multi-Tenancy World



Secret Chipmunk
Ron Parker | @scmunk

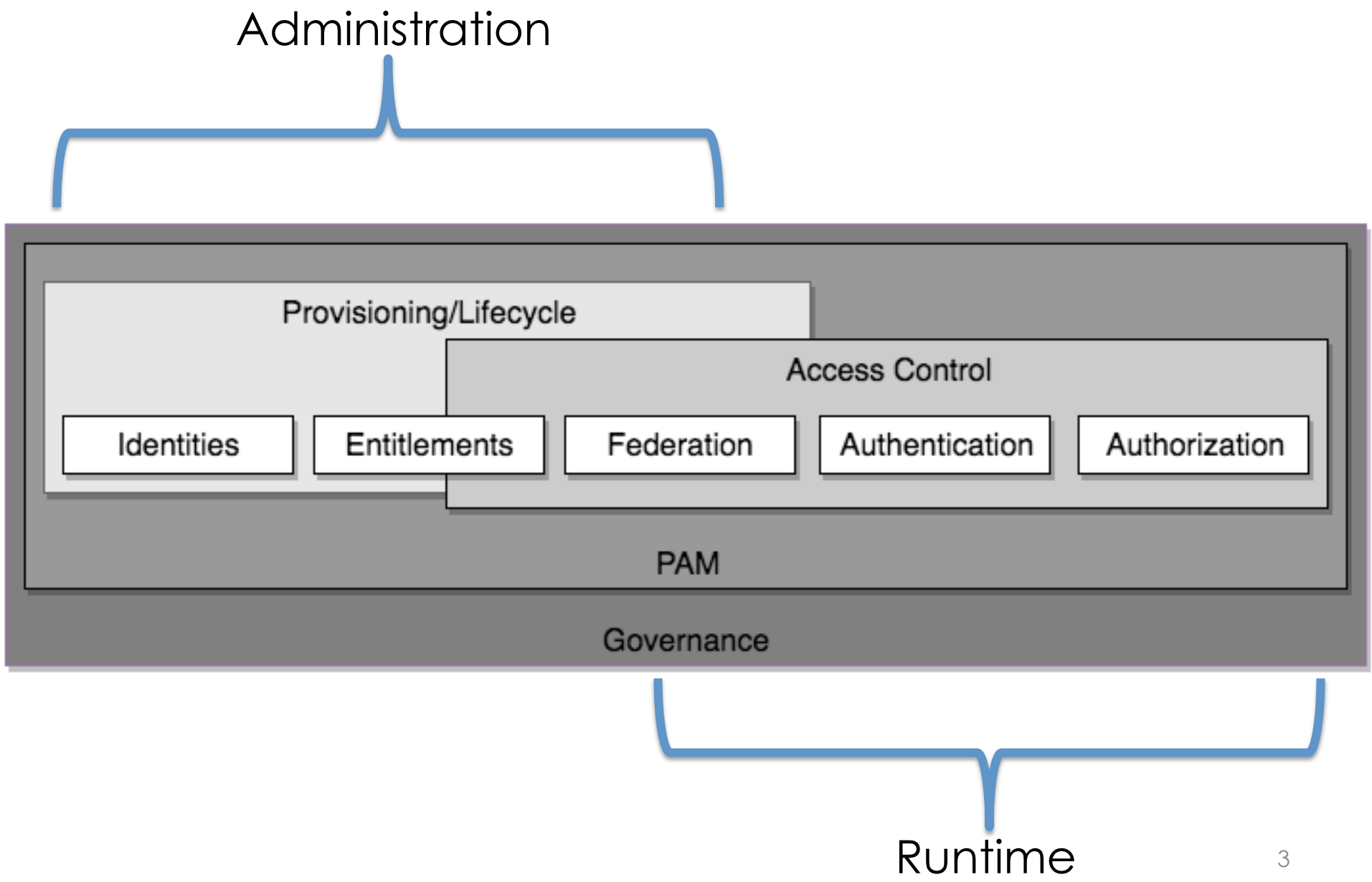
But what does that even mean?

We are in a super-shared, and super-scaled world where rights are no longer concentrated in one place.

Your rights and permissions are living everywhere in the neighborhood.

They have left home.

The World of Access



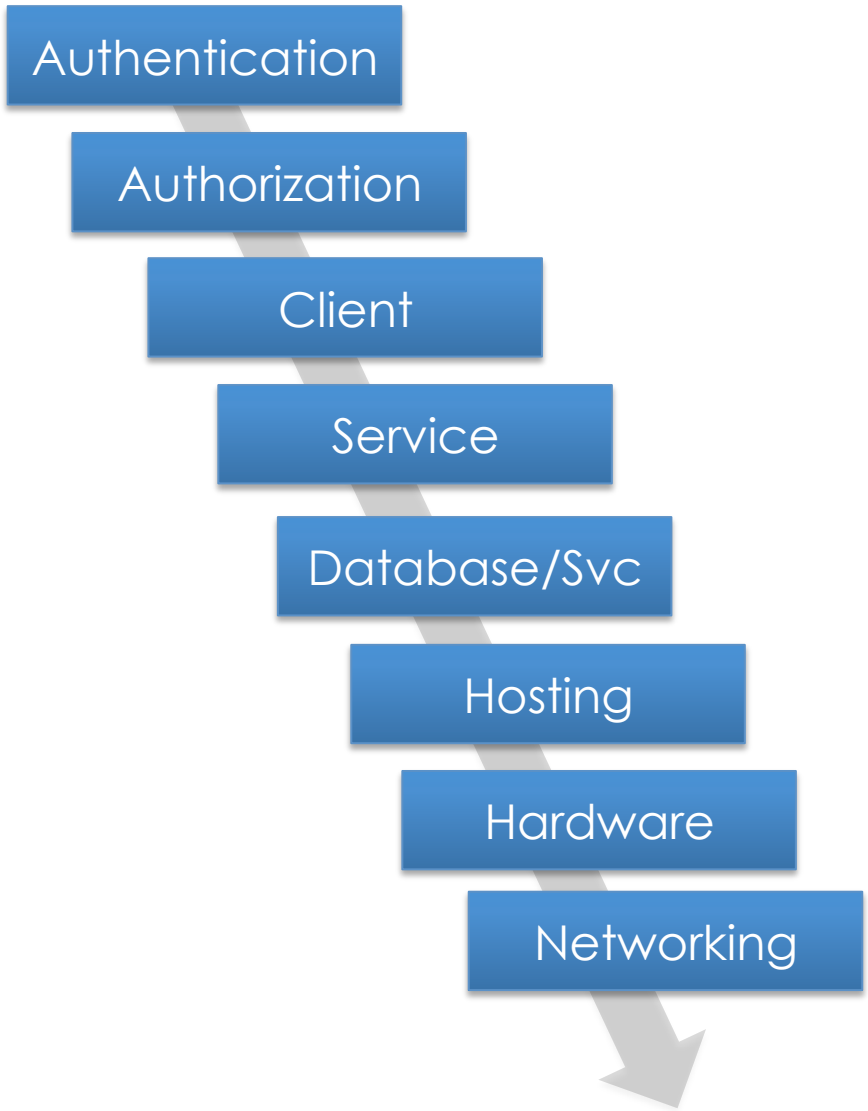
Entitlements

Do you have permission to do something

Can you be Authorized

- Created, engineered
- Granted and Provisioned
- Set up on the resource
- Maintained
- Audited

Entitlements are Everywhere



IDs/Pwds, Policies, hard-coded entitlements, multi-factor

Client Types, Serverless Apps, Old Apps, FaaS

Configurations, Unstructured Data, Structured, Queries

ACLs, Local Groups, GPOs

Identity Management, Virtualization, Containers, IaaS

More ACLs, Network Security Groups (NSG)

Entitlements

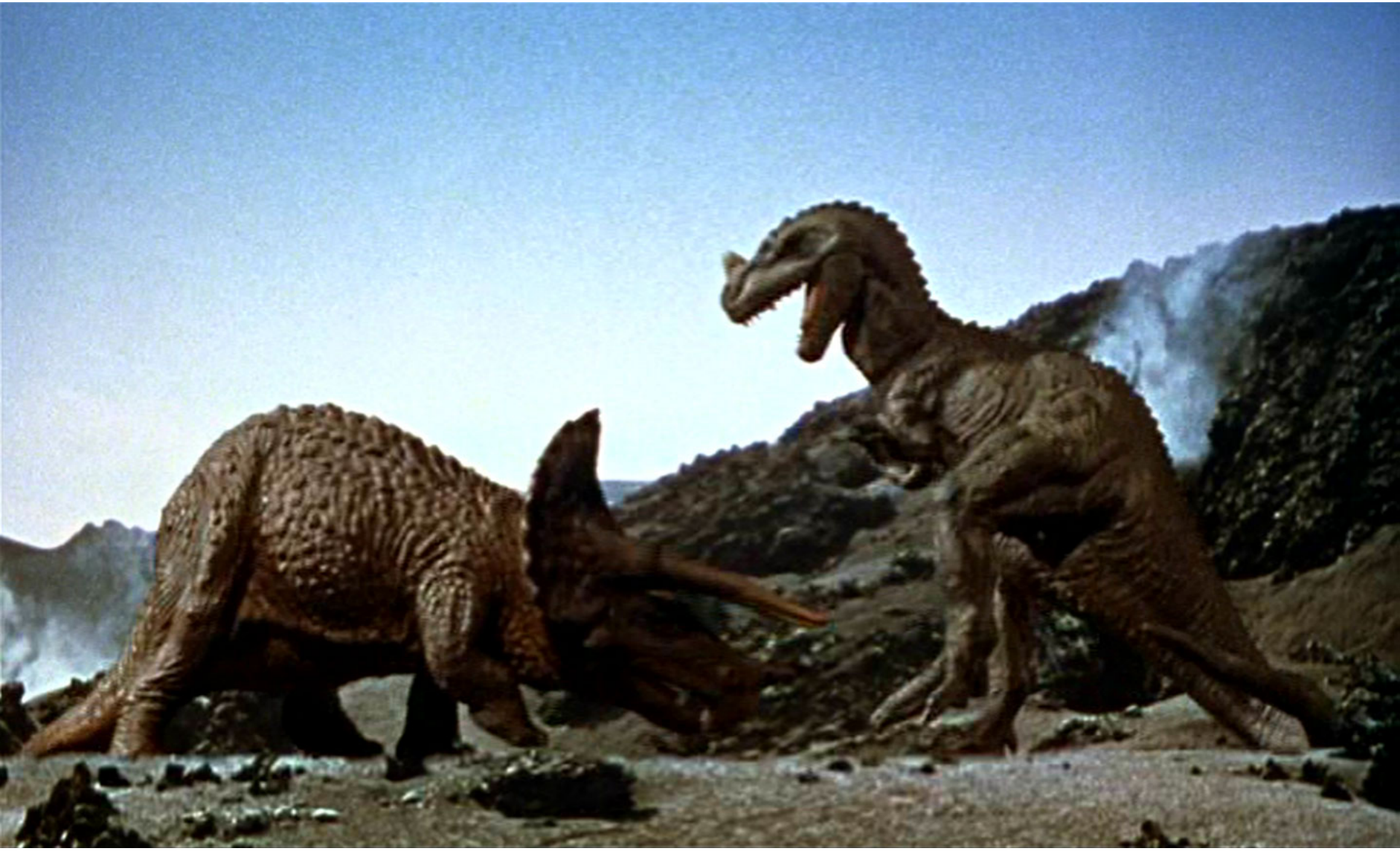
This works well with simple groups of accounts and resources that are all related.

The problem comes in when you want to operate on groups of accounts or resources that are not really related nor have the same owner.

This takes us into Multi-Tenancy.

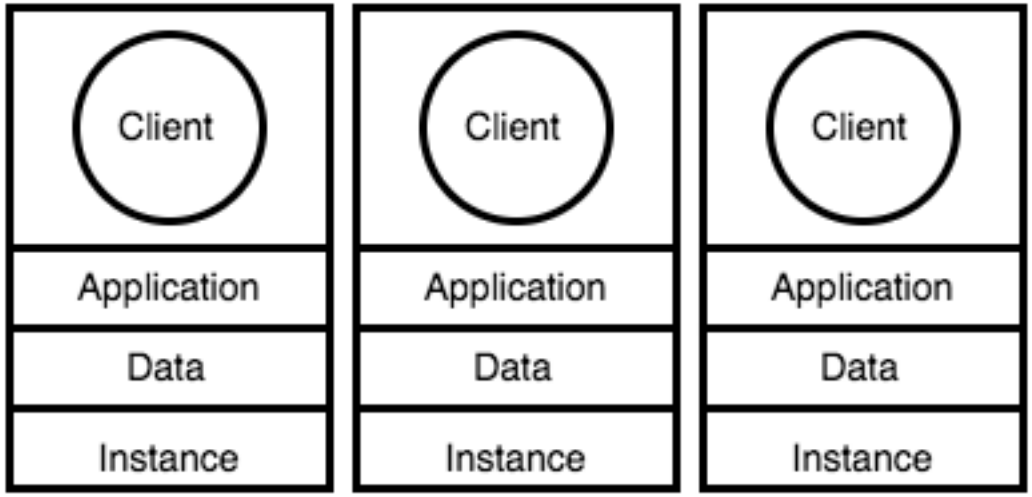
What is it really?

Haven't we been doing this forever anyway?



Multi-Tenancy

In the beginning we needed to serve multiple customers. We called these customer **tenants**. Each Tenant wanted to view or use the service as if it was their own.



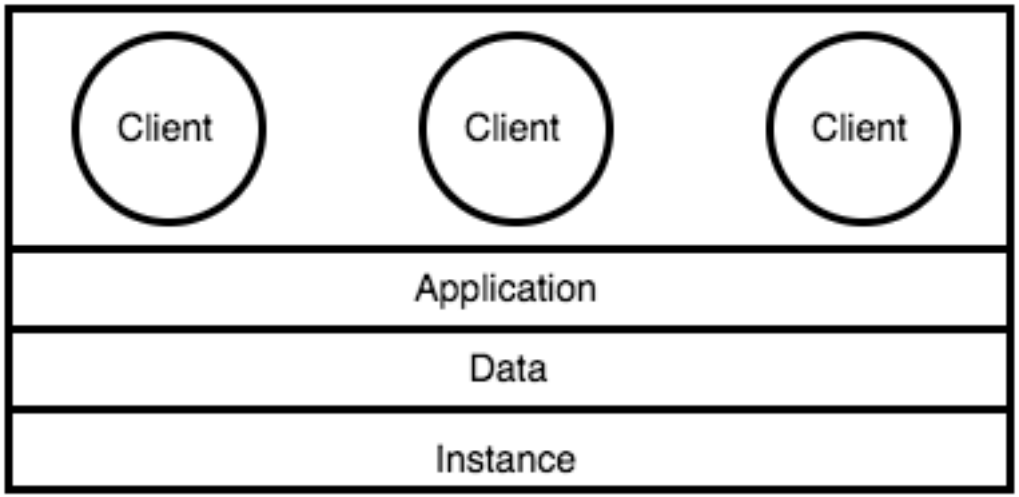
Basic Multi-Instance

Multi-Tenancy

We figured out ways to make it cheaper. We began to share more of the underlying service resources.

Tenants were sharing resources.
True Multi-tenancy.

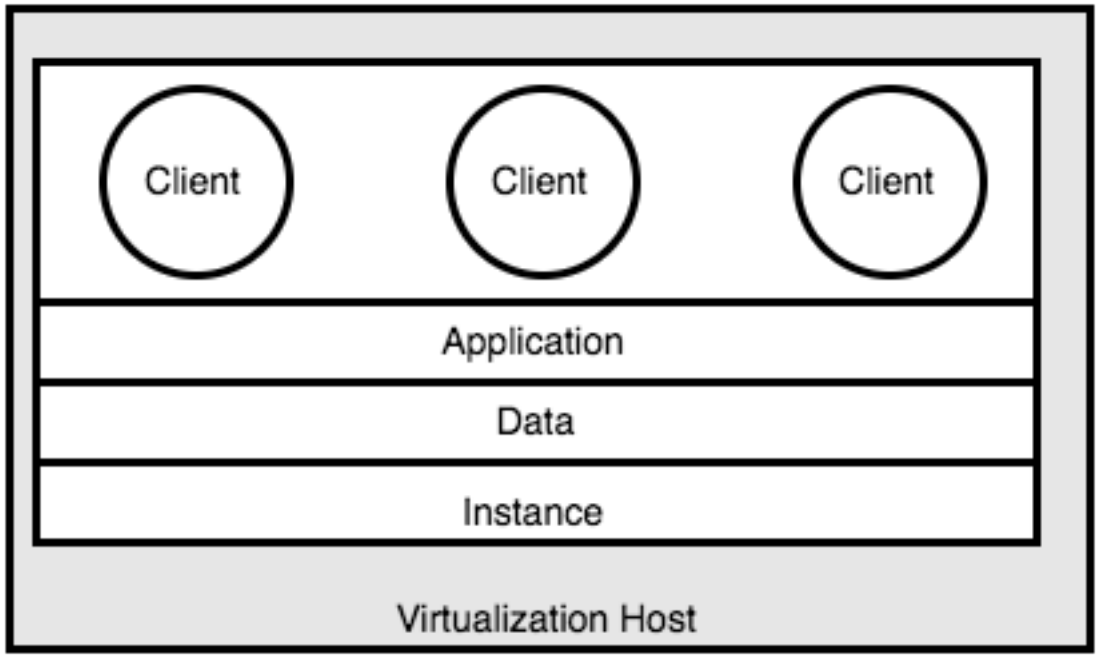
Scaling was still hard.



Basic Multi-Tenancy

Multi-Tenancy

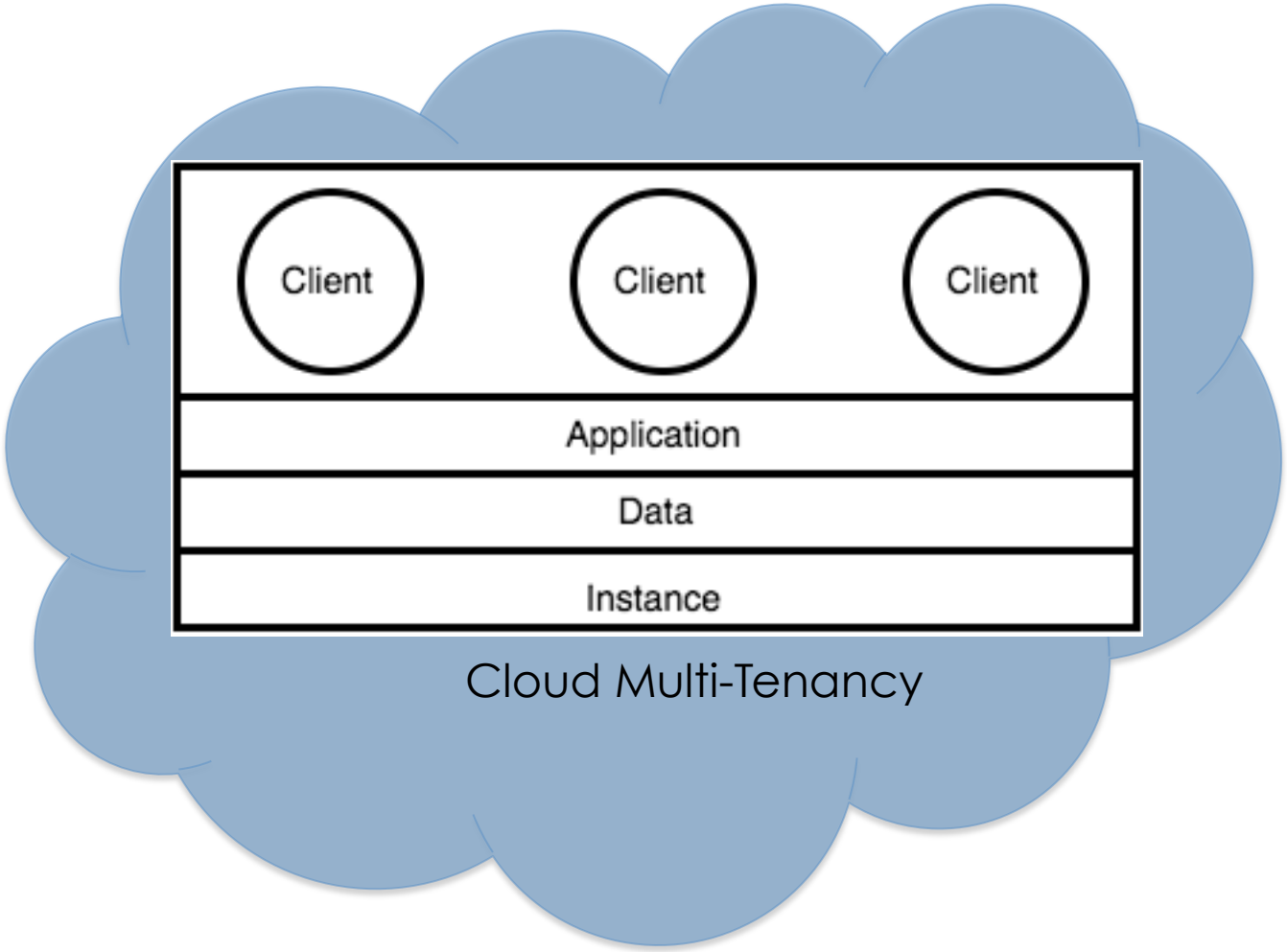
Eventually we even solved scaling.



Virtualized Multi-Tenancy

Multi-Tenancy

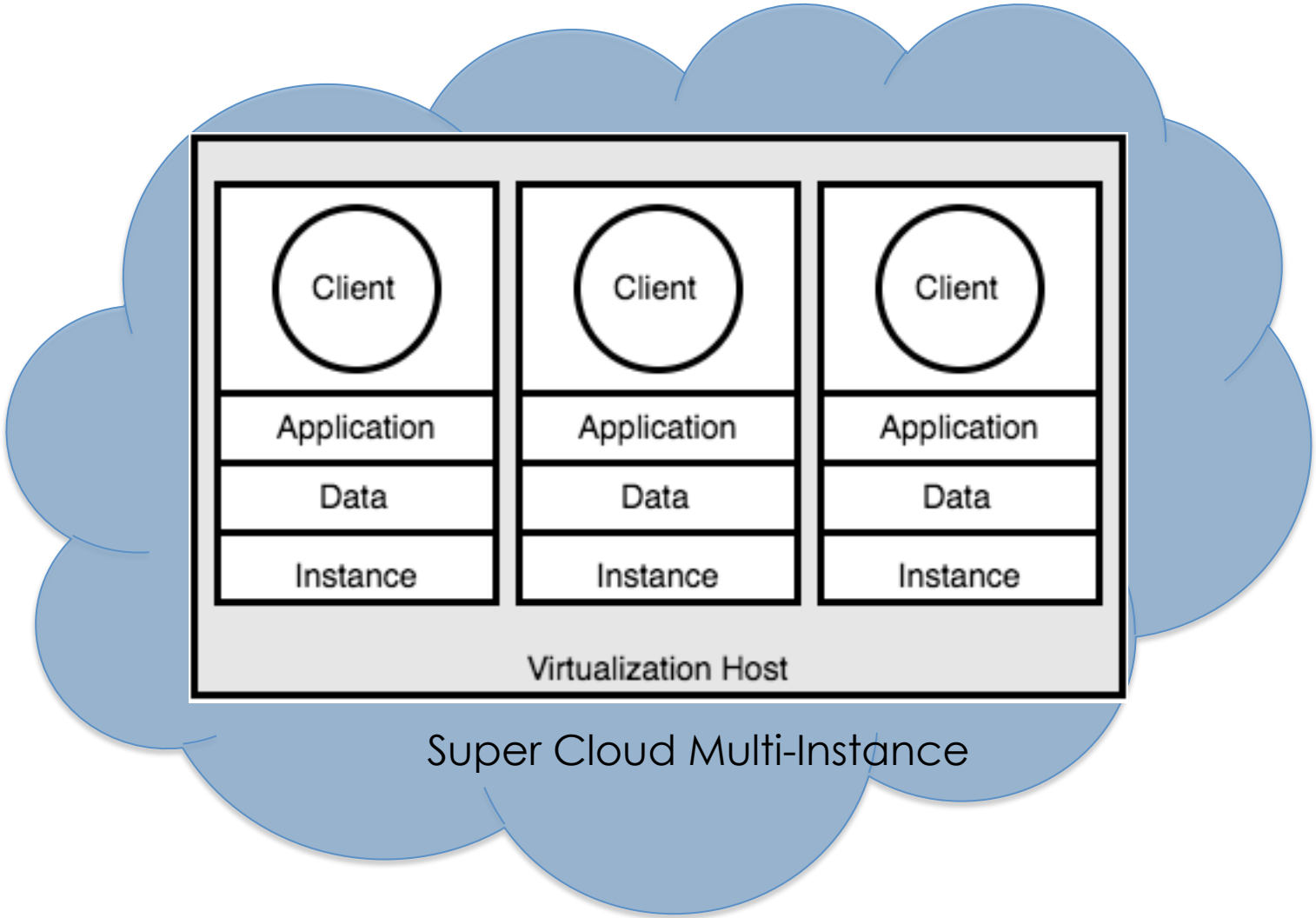
Then we moved to the cloud to let someone else take care of some of the issues.



Cloud Multi-Tenancy

Multi-Tenancy

Because scaling was there and it was inexpensive we could even go back to using multiple instances.



Multi-Tenancy and Entitlements

We still need authorizations.
All those permissions and rights.
All that for all of our tenants.

What is different when you put these two things together?

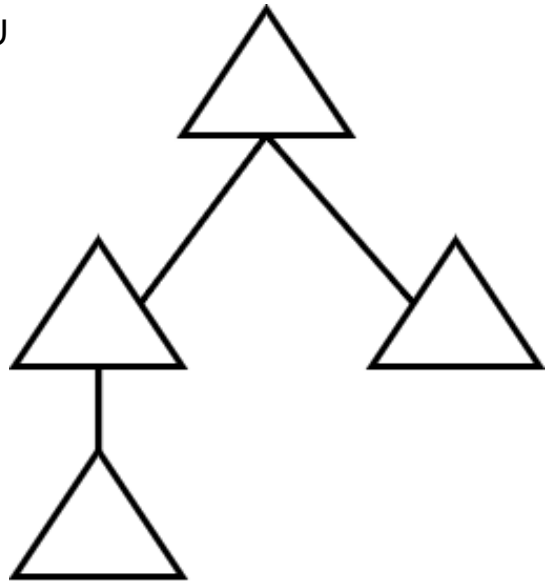
Issues

- Privacy
- Data Leakage
- Privilege Escalation
- Varying customer or regulatory requirements
- Privileged access management – groups of admins?
- Maybe you can't do Multi-tenancy for some parts of your data due to regulations

Examples

Active Directory/LDAP

- First Big Surprise – Active Directory is not MT-ready
- Admin entitlements and trusts are set up for a grou of admins
- You could make it work with separate account forests/domains and one-way trusts
- OpenLDAP and others are about the same
- So if you store all of you entitlements in LDAP you have to really think about the ramifications
- Doing this yourself can be dangerous

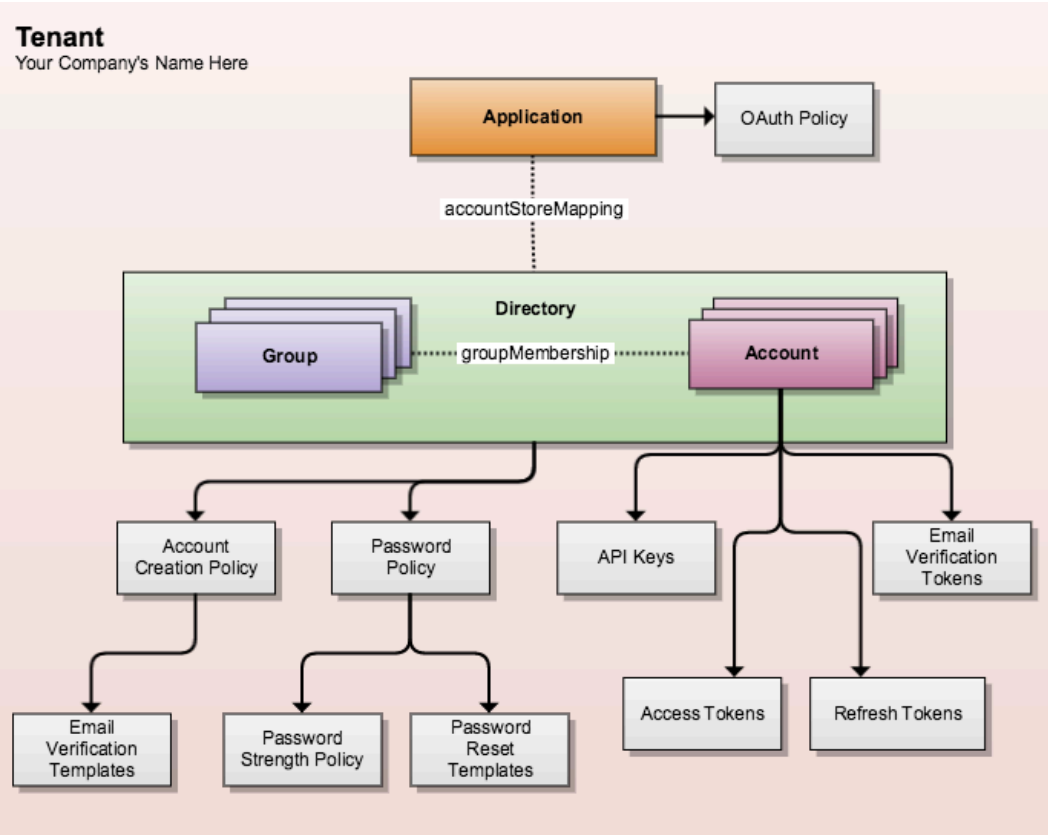


This is a huge reason that we have MS Azure AD that is much more multi-tenant aware.

StormPath – Vendor Example

- Cloud-ready service providing authentication, authorization, and account management
- Application centric with Accounts, Attributes, Groups
- REST Driven
- http://www.stormpath.com

Security Model



User REST Response

```
{
  "href": "https://api.stormpath.com/v1/
accounts/3apenYvL0Z9v9spExAMpLe",
  "username" : "jlpicard",
  "email" : "capt@enterprise.com",
  "givenName" : "Jean-Luc",
  "surname" : "Picard",
  "customData": {
    "permissions": {
      "crew_quarters": " 9-3601",
      "lock_override": "all",
      "command_bridge": {
        "type": "vessel:bridge",
        "identifier": "NCC-1701-D",
        "action": "lockout"
      }}}}
```

Databases

- There multiple approaches
  - Database per tenant
  - Single database with multiple tables per tenant
  - A Tenant ID in a shared table
  - Sharding
  - Policy

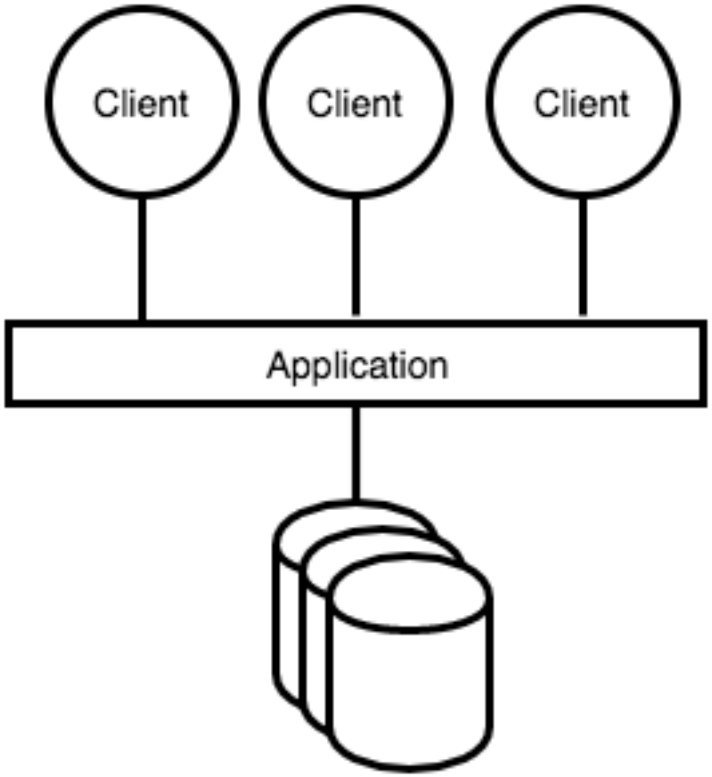
Shared Table Query

```
SELECT acct.name, acct.sales
      FROM acct.TID
      WHERE acct.TID = &1
```

Problem is you have to remember to do this for every query, stored procedure, view, or any other database object.

More Modern Databases - Sharding

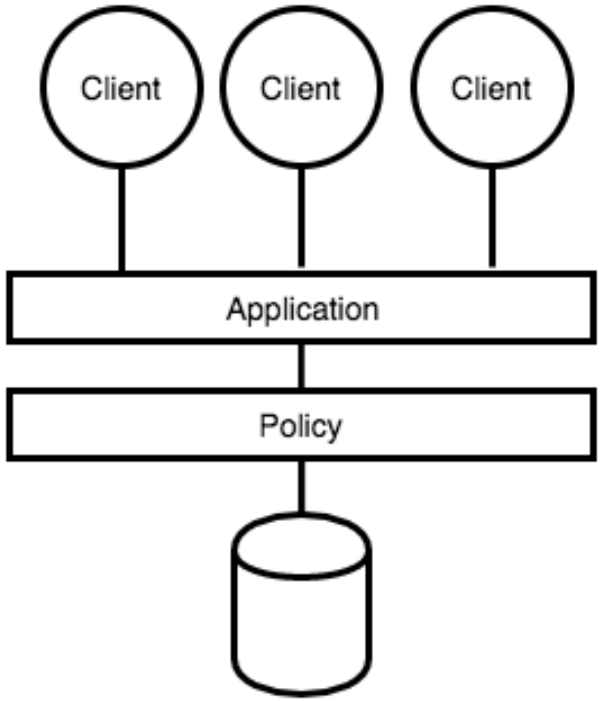
- Sharding – Separating data into logical, physical/geographical based on a key
- Many databases will shard based on a key



You can implement tenancy if the key is based n the tenant.

Databases – Policy-based Tenancy

- Some databases can use a policy to enforce tenancy
- MS SQL 2016 and AWS Dynamo
- You don't have to remember to "fix" each object
- Policy is system wide

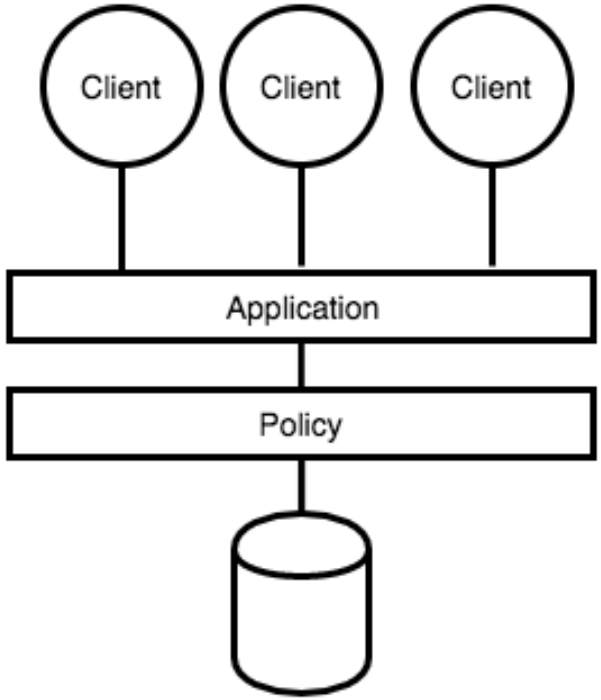


PREDICATE Function for MS SQL

```
 …WHERE
         (DATABASE_PRINCIPAL_ID() =
DATABASE_PRINCIPAL_ID('dbo') -- note, should not be dbo!
             AND CAST(SESSION_CONTEXT(N'TenantId') AS
int) = @TenantId)
         OR (DATABASE_PRINCIPAL_ID() =
               DATABASE_PRINCIPAL_ID('superuser'))
```

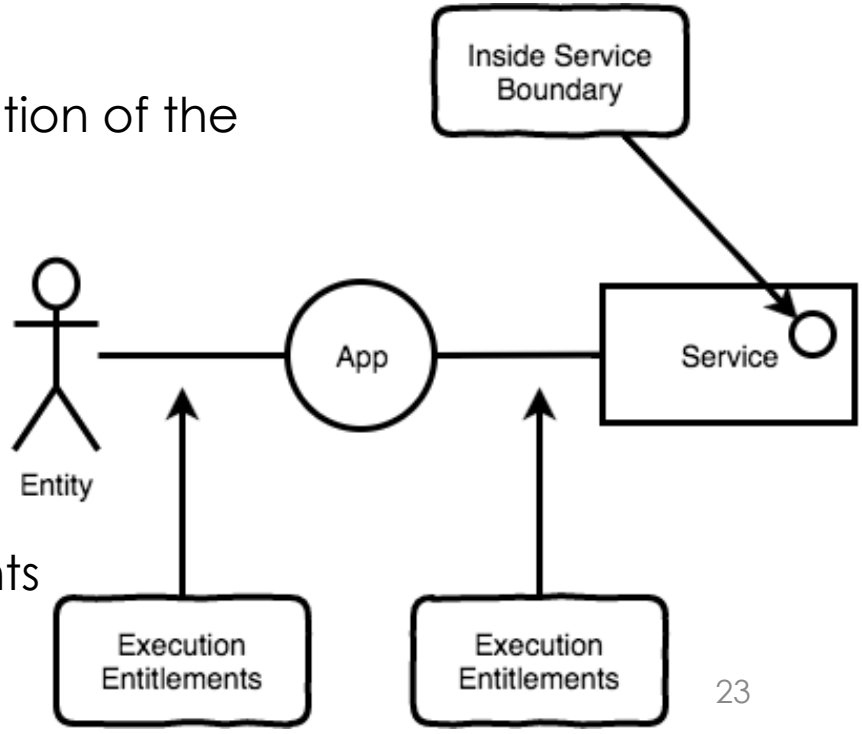
Reporting

- Don't forget any fixed reporting needs to take MT into account
- It is much harder to control dynamic reporting
- You want to look back at database level policies



Application Level

- You have to authenticate then capture the tenant information in a secure manner
- In a browser-based situation this mean you need a token or a session
  - A signed SAML token with a tenant assertion
  - Delegated authorization token with Oauth/JWT
  - Session cookies can be used but they are not ideal
- Oauth can be helpful using a combination of the SCOPE and the actual caller ID
- Authorization may be needed at the service boundary and later on inside the actual service
- All these components have to be MT aware in order to keep your entitlements straight



Backup and Recovery

- What are the ramifications of backing up all the tenants data to one spot, one file, one virtual tape?
- Do I have encryption or key issues?
- Is my Disaster Recovery solution also multi-tenant ready?

Summary

- Using MT vs Providing – most don't need to provide MT
- Building a MT environment is
  - Dangerous
  - Expensive
  - Needs to be built from the beginning
- MT is very hard out-of-the-box with older tools
- Most cloud-based services are built with MT in mind
- Maybe the customer doesn't care but the service provider has to

If someone says there is no difference between cloud and non-cloud computing you can bring up Multi-Tenancy.

It is a lot more than a deployment model.

Entitlements

In a Multi-Tenancy World

Thanks!

Secret Chipmunk

Ron Parker
@scmunk
http://www.secretchipmunk.com