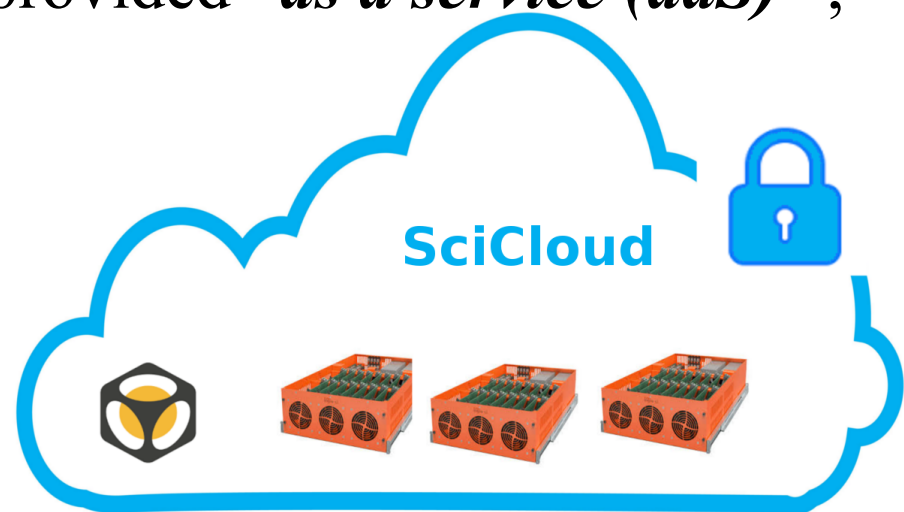


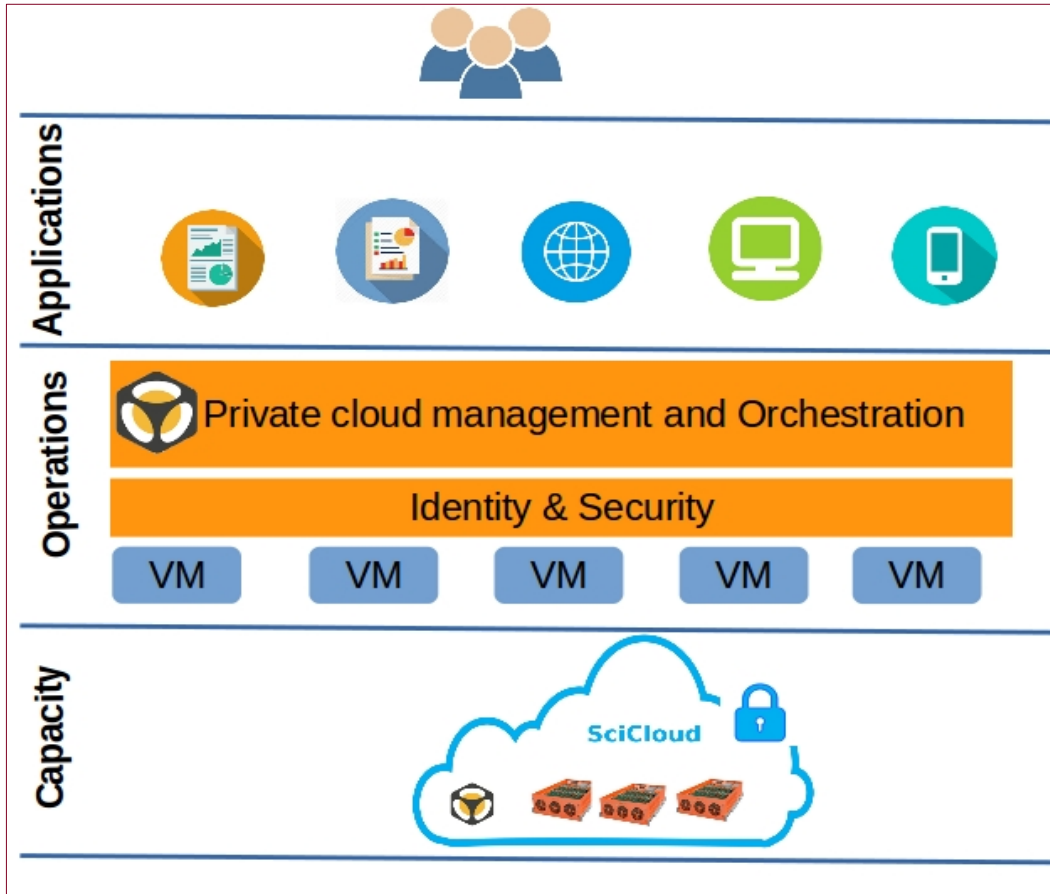
What is Scientific Cloud (SciCloud)?

- **SciCloud** is a private cloud for scientific computing for researchers
- It is running:
 - ✓ 18 physical servers
 - ✓ 80 cores
 - ✓ 1/5 TB of memory (RAM)
 - ✓ 12 TB storage (including backup storage)
- Provides customized images
- Real-time scalable resources provided “*as a service (aaS)*” ,
e.g.:
 - ✓ Data (DaaS)
 - ✓ Analytics (*AaaS*)



SciCloud Overview

Architecture



Web-based admin console

The screenshot shows the web-based admin console for SciCloud. The browser address bar displays <https://engine1.sciencecloud.dk/steamengine/mainvalve/>. The console has a top navigation bar with tabs: Dashboard, Servers, Images, and Connections. Below this is a sub-navigation bar with tabs: Info, Usage, Monitoring, Software, and Graphs. The main content area is divided into two sections:

- Account:** A sidebar on the right containing user information:
 - Owner name: Xiufeng Liu
 - Owner email: xiuli@dtu.dk
 - Owner phone: 93511339
 - Technician name: [edit]
 - Technician email: [edit]
 - Technician phone: [edit]
 - Alert email: [edit]
 - Allow login from: [edit]
 - Last login: [edit]
 - Last login from: [edit]
 - Unauthenticated API access: ☒
 - Actions: [reset all contacts](#)
- All apps:** A table listing running applications:

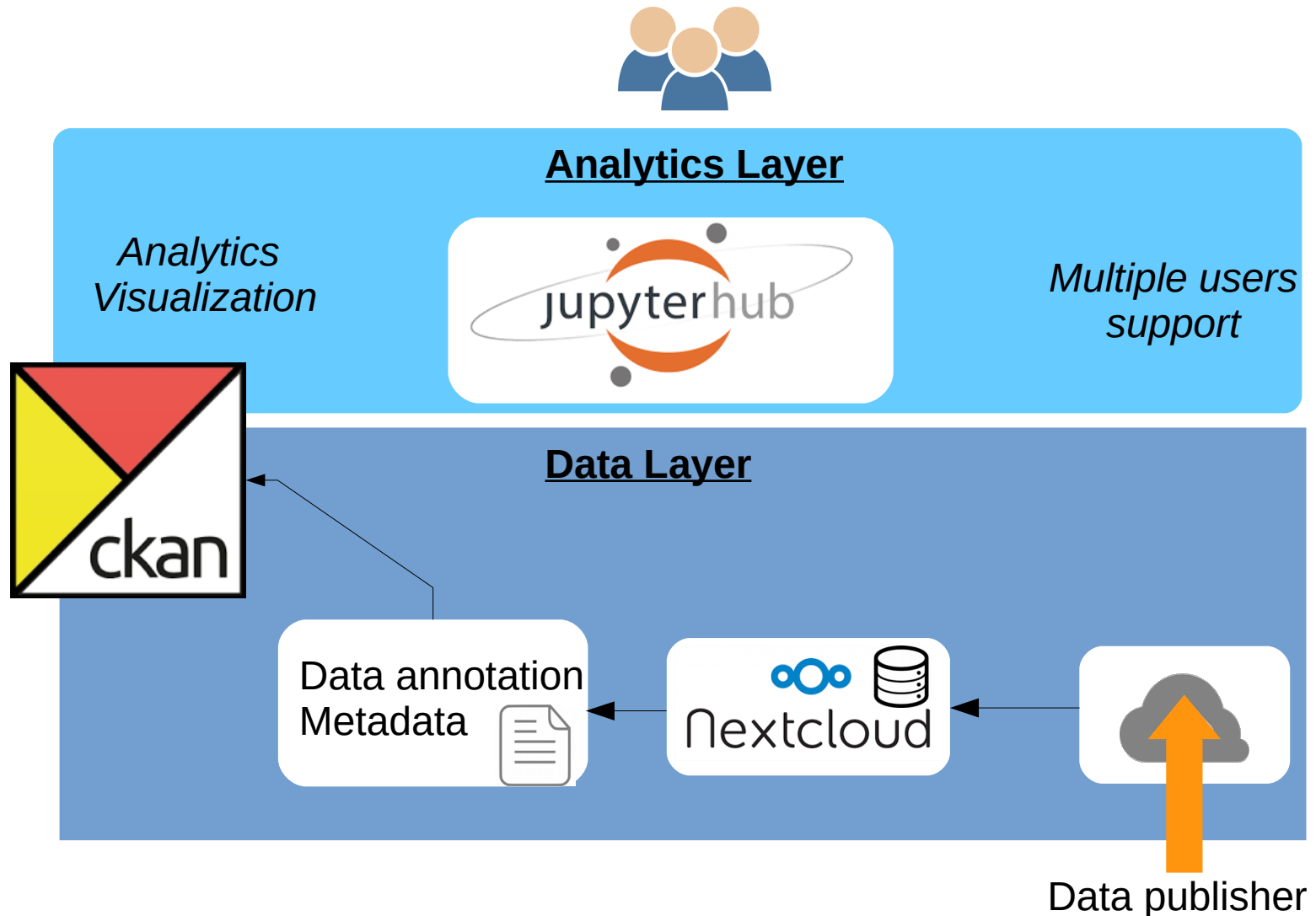
Name	Status	Action
Jupyterhub	running	[stop] [restart] [refresh] [delete]
SDU_Rune	running	[stop] [restart] [refresh] [delete]
CTT	running	[stop] [restart] [refresh] [delete]

Below the table, a summary of system resources is provided:

- Total servers: 3 (12 vCPUs, 24576 MB memory)
- Active servers: 3 (12 vCPUs, 24576 MB memory)
- Images: 3 (3 active)
- Connections: 3

An "Activity log" section is visible at the bottom of the console.

SciCloud – *Analytics as a Service (Jupyterhub)*



SciCloud – Analytics as a Service (Jupyterhub)

The image illustrates the JupyterHub interface through three sequential screenshots, connected by green curved arrows indicating the workflow.

Left Screenshot: Jupyter Hub Login
The browser shows the Jupyter Hub login page. A warning message states: "Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub." Below the warning, the login form includes:
- Username:
- Password:
- A "Sign In" button.

Middle Screenshot: Jupyter File Manager
After login, the user is in the Jupyter file manager. The "Files" tab is active, showing a list of items:
- Example:notebooks
- Example:Python for Data Science
- notebooks
- DataSets.ipynb
- Untitled.ipynb
A context menu is open over the "Untitled.ipynb" file, showing options: Text File, Folder, Terminal, Notebooks, Python 2, Python 3, and R.

Right Screenshot: Jupyter Notebook Execution
The user is in a Jupyter notebook titled "13.Visualization". The code in the cell is:

```
In [12]: # set up hist and plot it
bins = np.linspace(-3, 3, 50)
plt.hist(samples_1, bins=bins, alpha=0.5, label='samples 1')
plt.hist(samples_2, bins=bins, alpha=0.5, label='samples 2')

# Legend
plt.legend(loc='upper left')
```

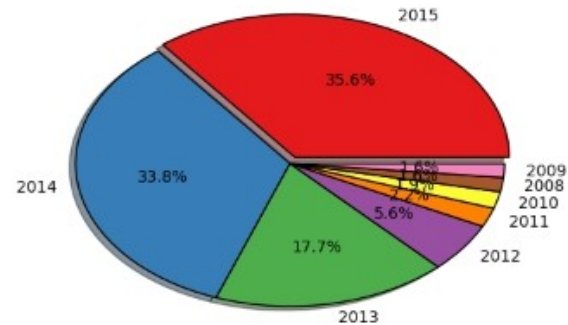

The output shows a histogram with two overlapping distributions: "samples 1" (blue) and "samples 2" (green). The x-axis ranges from -3 to 3, and the y-axis (frequency) ranges from 0 to 1000. The legend is located in the upper left corner of the plot area.

0. Description of the data

The data used is the district heating consumption of single family houses in Aarhus. The data contains the consumption of customers at 1 hour intervals. There are 15,065 installations and the total number of 24 hour load profiles is 9,051,636. The data ranges from December 2008 to November 2015. However, the data range is different depending on the installations. For example, there are a small number of installations having the data before 2013. There are 7,507 installations with data ranging in 2013, and 14,333 in 2014, and 15061 in 2015. We use the data of 2015 in this experiment.

```
In [98]: dailyProfiles = [15061, 14333, 7507, 2384, 920, 805, 681, 663]
Years = [2015, 2014, 2013, 2012, 2011, 2010, 2009, 2008]

fracs = [100.0*x/sum(dailyProfiles) for x in dailyProfiles]
explode=(0.05, 0, 0, 0, 0, 0, 0, 0)
pie(fracs, explode=explode, labels=Years, colors=cm.Set1(np.arange(8)/8.0), autopct='%1.1f%%', shadow=True);
```



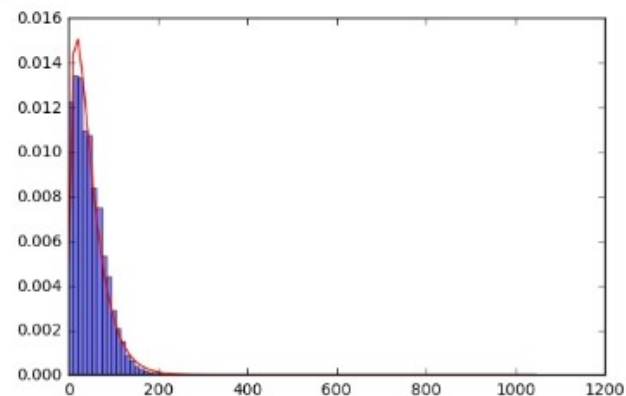
In the following, we use histogram to plot the distribution of daily profiles of 2015, and we see that the gamma distribution can fit the best.

```
In [73]: X = np.load('HourlyReadings_2015.npy')
```

```
In [99]: dailySum = X.sum(axis=1)
x = np.linspace(dailySum.min(), dailySum.max(), 100)

# fit
param = stats.gamma.fit(dailySum)
pdf_fitted = stats.gamma.pdf(x, *param)
plt.plot(x, pdf_fitted, color='r')

# plot the histogram
plt.hist(dailySum, normed=True, bins=100, alpha=0.5);
```

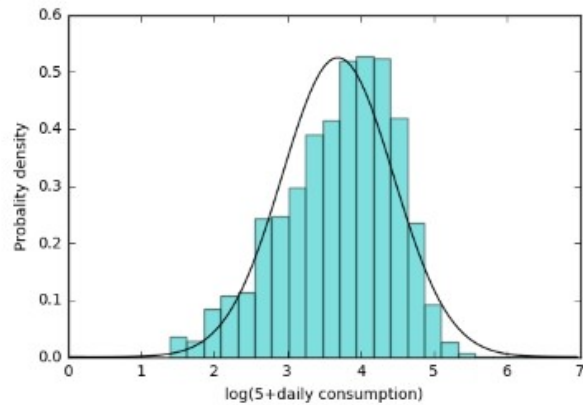


The mixture of log normal distributions fits best the actual data.

$$f(a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\log(a+\epsilon)-\mu)^2}{2\sigma^2}} \text{ where } n = 0, 1, \dots$$

```
In [139]: logdata = np.log(5+dailySum)
logdata = logdata[~np.isneginf(logdata)]
estimated_mu, estimated_sigma = stats.norm.fit(logdata)
xmin = logdata.min()
xmax = logdata.max()

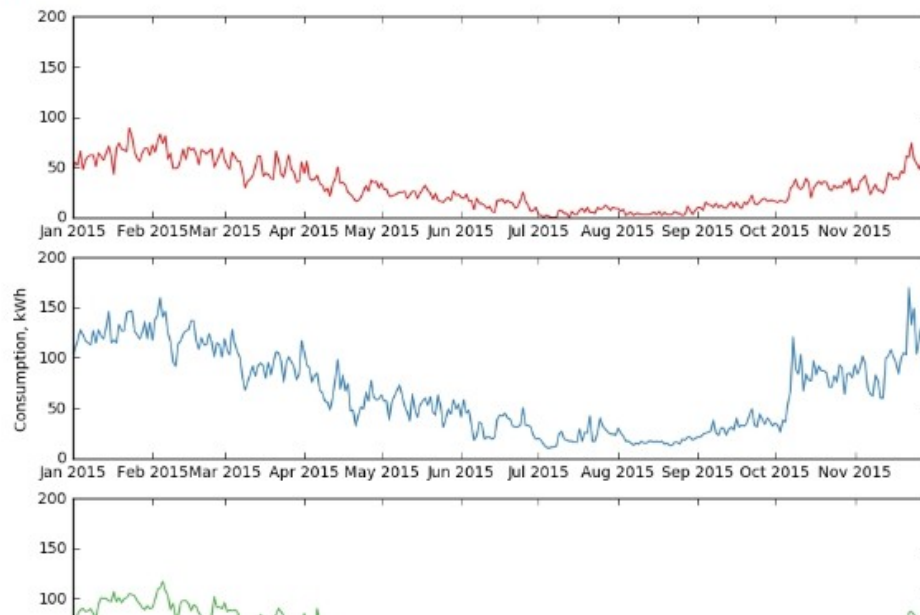
plt.hist(logdata, bins=30, normed=True, color='c', alpha=0.5, range=(0, xmax))
x = np.linspace(0, xmax, 100)
pdf = stats.norm.pdf(x, loc=estimated_mu, scale=estimated_sigma)
plt.plot(x, pdf, 'k')
plt.xlabel("log(5+daily consumption)")
plt.ylabel("Probability density");
```



```
In [146]: df = pd.read_csv('ava_dailydata_2015.csv', header=0, sep='|')
```

```
In [175]: fig, axs = plt.subplots(nrows=3, ncols=1, figsize=(10, 8))
nos = [100015, 100020, 100166]
colors = cm.Set1(np.arange(3)/8.0)

for i in range(3):
    ts = df[df.installnr==nos[i]]
    ts.index = pd.to_datetime(ts['readdate'])
    axs[i].plot(ts.index, ts.reading, color=colors[i])
    axs[i].set_ylim([0, 200])
#axs[2].set_xlabel('Month')
axs[1].set_ylabel('Consumption, kWh')
plt.plot();
```

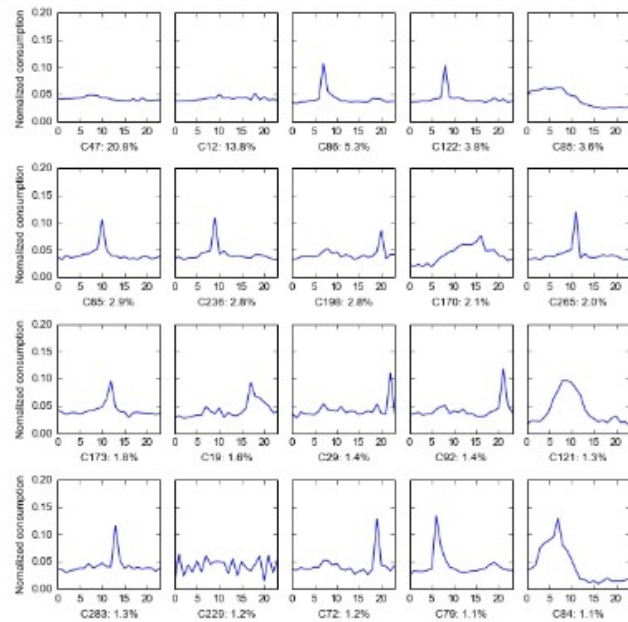


3. Clustering

We adopt the two-stage clustering method proposed in [1], first by adaptive clustering following the merging small clusters.

```
In [ ]: km = adaptive_clustering(sampleX, 10, 1000, 0.3)
labels, centroids = hierarchical_clustering(np.copy(km.labels_), np.copy(km.cluster_centers_))
```

```
In [9]: plot_centroids(labels, centroids, 4, 5, 0.2)
```



```
In [8]: plot_points(sampleX, labels, centroids, 4, 5, 0.2)
```

