Workspace & Data

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

BUILD THE FOUNDATION

- **Workspace** — Organize
- **Identity** — Authorize
- **Data** — Reference
- **Compute** — Execute

01 Design the workspace and data foundation

Workspace and development interfaces

- A **workspace** organizes jobs, experiments, data, models, environments and compute. Associated Storage, Key Vault, Container Registry and monitoring resources serve distinct storage, secret, image and telemetry roles.
- **SDK v2** uses **azure.ai.ml**, **MLClient** and credential objects such as **DefaultAzureCredential**. Identify the subscription, resource group and workspace; configuration files identify resources and do not replace authentication.
- **SDK v1** vocabulary includes **azureml.core**, **Workspace.from_config()**, **Experiment**, **Run** and **ScriptRunConfig**. Keep these APIs separate from v2 command jobs and MLflow tracking.
- Use **studio notebooks** for interactive code, **designer** for visual pipelines, and the **CLI ml extension** for scripted resource and job operations. CLI v2 uses YAML definitions; an AmlCompute definition uses the **amlCompute** schema.

Match compute to the workload

- A **compute instance** is an interactive development workstation. An **AmlCompute cluster** provides shared, elastic capacity for training and batch scoring. An online deployment needs an inference hosting configuration.
- The **submission machine** and **training target** can differ: a CPU notebook can submit a CUDA training script to GPU compute. **CUDA** requires compatible NVIDIA hardware, drivers and framework libraries.
- A **Data Science Virtual Machine** supplies a preconfigured data science environment. GPU resources accelerate suitable deep learning workloads; geospatial tooling serves GIS work. Select hardware and dependencies for the actual workload.
- **Kubernetes compute** can support training and inference on configured cloud or Azure Arc connected infrastructure. An **instance_type** selects an administrator-defined resource profile; it is distinct from the compute target name.

Scaling, provisioning and cost

- Cluster **min_instances** controls idle capacity; **max_instances** caps scale. A minimum of zero and an idle scale-down timeout reduce idle cost while introducing startup latency. Quotas and regional capacity can limit provisioning.
- **Dedicated** nodes avoid Spot eviction risk. **Low-priority/Spot** nodes reduce cost for interruption-tolerant work; checkpoint models and make retries safe. Changing the maximum changes the available capacity ceiling.

Workspace & Data

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

BUILD THE FOUNDATION

- **Workspace** Organize
- **Identity** Authorize
- **Data** Reference
- **Compute** Execute

- Use **ml_client.compute.begin_create_or_update()** to start asynchronous provisioning and a poller's **result()** when later work requires completion. Retrieving an existing target differs from creating a new one.
- **Serverless training** allocates job compute without managing a persistent cluster. **ResourceConfiguration** specifies requested instance type and count. Compute-instance schedules control start/stop times; cluster scaling follows job demand.

Spark and data engineering

- **Spark** distributes data processing across workers. **Databricks** and **Synapse Spark** support notebook-based analytics; **Data Factory** orchestrates movement and processing across services.
- Attach a Synapse pool with **SynapseSparkCompute**, a workspace compute name and the pool's **resource ID**. Compute registration, session startup and storage authorization are separate operations.
- **Serverless Spark** suits interactive wrangling and large-scale transformations. Dynamic executor allocation adjusts resources; keeping a session warm can reduce repeat startup delay at an additional idle cost.
- A **system-assigned identity** follows its resource lifecycle. A **user-assigned identity** is independently managed and reusable. Configure the compute identity and grant it the required storage permissions before processing data.

Datastores, data assets and formats

- A **datastore** registers a connection to storage; registration does not copy the data. A **data asset** gives a reusable name and version to a data reference. Neither alone freezes mutable files at the referenced location.
- In SDK v2, **uri_file** references one file, **uri_folder** a collection, and **MLTable** a tabular loading blueprint. **mltable.load()** reads an MLTable definition; **from_delimited_files()** and **from_parquet_files()** create tables from files.
- In SDK v1, **FileDataset** exposes files for mount/download; **TabularDataset** parses records and supports **to_pandas_dataframe()**. Choose by file access versus tabular interpretation, rather than whether data is already clean.
- **MLTable** supports lazy transformations such as column selection before materialization. A **CSV/TSV** is delimited text; **Parquet** is columnar; **ARFF** describes attributes and data for Weka.
- Use immutable storage paths and explicit file lists for reproducible cutoffs. A wildcard may include files added later. Version the data reference and preserve the referenced bytes when historical reproduction matters.

Workspace & Data

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

BUILD THE FOUNDATION

- **Workspace** — Organize
- **Identity** — Authorize
- **Data** — Reference
- **Compute** — Execute

Storage access and workspace security

- **AzCopy**, **Storage Explorer** and storage SDKs move data to Blob Storage. Register existing storage to avoid unnecessary copies. ADLS Gen2 uses **abfss://filesystem@account.dfs.core.windows.net/path** for secure filesystem access.
- **Credential-less datastores** use an authorized user or job identity. Stored credentials can use a service principal, account key or supported SAS. **SAS** limits permissions, scope and expiry; account keys require deliberate rotation.
- Grant **storage data roles** and, where applicable, ADLS directory **ACLs**. A workspace role does not automatically authorize underlying data. Studio preview, interactive notebooks and remote jobs can use different identities.
- **Private endpoints** require reachable networks and correct DNS as well as authorization. Custom **RBAC** roles use Actions and NotActions at defined scopes; NotActions subtracts permissions from that role and is not a deny against other role grants.
- Use **managed identities** or service principals for unattended workflows. The workspace's **hbi_workspace** setting adjusts sensitive diagnostic handling; it does not itself provide differential privacy or replace a complete security design.

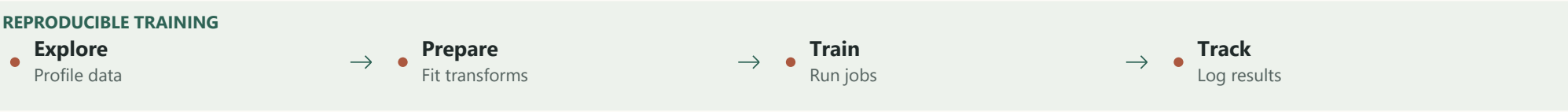
Notebook kernels and source control

- Notebook variables live in the **kernel process**. Restarting the kernel or stopping compute clears in-memory state. Merely navigating away does not reset it. **%pip** targets the active Python kernel's environment.
- Create a separate environment, install dependencies and **ipykernel**, then register it with **python -m ipykernel install**. Use **jupyter kernelspec list/uninstall** to manage kernels; R kernels use **IRkernel** registration.
- For **Git over SSH**, generate a key pair, add the **public key** to the Git provider, protect the private key, then clone the SSH URL. Choose the intended workspace or local filesystem for the repository.
- An **.amlignore** file takes precedence over **.gitignore** for Azure ML source uploads. Keep large data and secrets out of code snapshots. Track the **Git commit**, branch, environment and data versions with each run.
- Refactor notebook logic into reusable functions and a script with a guarded **main()** entry point. A compatible local **Docker** host supports container testing; it needs enabled virtualization and an appropriate container backend.

Training

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET



02 Explore data and train models

Profile and reshape data

- Use **df.describe()** for numeric count, mean, standard deviation, extrema and quartiles; **df.isna().sum()** counts missing values by column. Review types, unique values, parse errors and distributions before modeling.
- **Positive skew** has a longer right tail; **negative skew** a longer left tail. Histograms and Q-Q plots reveal shape; a mean or standard deviation alone cannot establish normality. Box plots and scatter plots help locate outliers.
- **Add Rows** or **pandas.concat()** appends compatible rows. **Join Data** combines columns through keys. **pandas.melt()** unpivots wide columns: **id_vars** stay as identifiers and **value_vars** become long-form observations.
- **Edit Metadata** adjusts types and categorical treatment. **Apply SQL Transformation** or **Execute Python Script** computes new features. Designer's Python entry point is **azureml_main** and returns supported dataframe outputs; bundle helper code and libraries as needed.

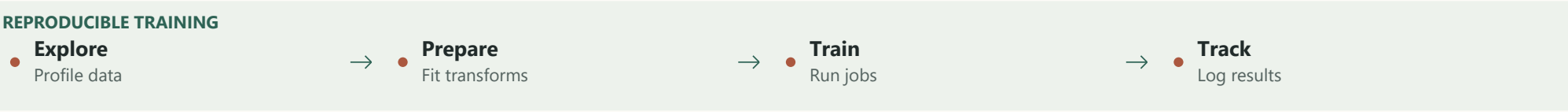
Missing values, duplicates and outliers

- **Mean/median** imputation suits numeric features; median is less sensitive to extreme values. **Mode** suits categories; a custom constant can mark unknowns. Fit imputation on training data and reuse it for validation and inference.
- **MICE** models missing variables conditionally on other variables. **Probabilistic PCA** uses a latent covariance structure. **Last observation carried forward** assumes temporal continuity and can bias results when that assumption fails.
- Deleting rows reduces sample size; deleting columns changes the feature set. Imputation can preserve dimensions, but entirely missing columns need an explicit policy. Investigate the missingness mechanism before choosing a treatment.
- Remove accidental **duplicates** without merging legitimate repeated events. Quantify **outliers** before removal; validate domain plausibility and consider robust scaling or **Clip Values** thresholds. Apply learned thresholds consistently to new data.

Training

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET



Encode, scale and extract features

- **One-hot encoding**, called **Convert to Indicator Values** in Studio modules, creates binary category columns. Use ordinal codes only when order has meaning; handle unseen categories explicitly.
- **StandardScaler** centers each feature and scales its variance; it does not make data Gaussian. **MinMaxScaler** maps a feature to a chosen range. **Normalizer** scales each row to unit norm. Fit feature scalers only on training data.
- **Equal-width bins** divide a numeric interval evenly; **quantile bins** target similar row counts. **Entropy MDL** is supervised binning using a target. Binning and continuous scaling solve different representation needs.
- Text preparation can lowercase, remove selected stop words and punctuation, and **lemmatize** to canonical words. **N-grams** capture token sequences; a maximum of three includes unigrams, bigrams and trigrams. Reuse the training vocabulary at inference.
- **TF-IDF** weights informative terms; **feature hashing** maps tokens into fixed-size buckets and can collide. **Latent Dirichlet Allocation** discovers text topics. Count-based transforms turn learned frequency tables into reusable features.

Splits, sampling and leakage

- Separate **training**, **validation** and final **test** roles. Tune on validation data and reserve the test set for final assessment. Fit preprocessing, feature selection and resampling inside each training fold to avoid leakage.
- **Stratification** preserves class proportions across splits; it does not rebalance the population. Use chronological validation for time series and group-aware splitting when related records must stay together. A fixed seed supports repeatability.
- **K-fold cross-validation** trains on k-1 folds and validates on the remaining fold, rotating through all folds. Five or ten folds are common tradeoffs; **leave-one-out** uses one observation per validation fold and can be expensive.
- **Partition and Sample** can sample rows or assign folds; sampling **with replacement** permits reuse. Designer's **Split Data** supports random rows or conditions: numeric/date expressions and text-pattern conditions partition by different rules.
- **SMOTE** synthesizes minority examples from nearby training cases. Oversampling, undersampling and **class weights** address imbalance; apply them only to training data and evaluate on a representative untouched distribution.

Training

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

REPRODUCIBLE TRAINING

Explore Profile data → **Prepare** Fit transforms → **Train** Run jobs → **Track** Log results

Feature selection and statistical tests

- **Pearson correlation** measures linear association; **Spearman** and **Kendall** measure rank association. Correlation near zero does not rule out nonlinear dependence. **Mutual information** can capture broader statistical dependence.
- **Chi-squared** tests categorical association; scikit-learn's chi-square selector expects nonnegative features. **ANOVA** compares group means. **Filter-based feature selection** ranks features before fitting the final predictor.
- **PCA** projects features onto variance-maximizing axes and can lose information. Fit PCA on training data, then transform validation/test data with that fitted object. **Fisher linear discriminant analysis** uses labels to seek class separation.
- A **paired t-test** analyzes within-pair differences; an unpaired test compares independent groups. Use a **two-sided** alternative for either direction. Paired observations should be independent across pairs, with suitable assumptions for the differences.

Choose the model family

- **Regression** predicts numeric outcomes; **classification** predicts categories. **Poisson regression** models nonnegative integer counts with an appropriate mean-variance assumption. Linear models are baselines; nonlinear models can capture interactions.
- **Decision forests** combine trees; **boosting** sequentially corrects residual errors. **Decision jungles** use ensembles of directed acyclic graphs. **Bayesian linear regression** expresses uncertainty through distributions over parameters.
- For **SVM/SVC**, **C** controls the penalty for training errors relative to regularization. **class_weight='balanced'** weights classes inversely to frequency. Scale features when distance or margin geometry matters.
- **K-means** assigns examples to nearby centroids and minimizes within-cluster squared distances. Centroid stability, tolerance or an iteration limit can stop training. Recommenders such as **Matchbox** relate users and items, optionally using side features.
- **CNNs** learn spatial patterns in images; **RNNs** model sequences; **GANs** train a generator against a discriminator. **TensorFlow/PyTorch** provide deep learning tools; **scikit-learn** supplies many tabular learning algorithms.

Training

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

REPRODUCIBLE TRAINING

Explore Profile data → **Prepare** Fit transforms → **Train** Run jobs → **Track** Log results

Training jobs and reproducible environments

- A v2 **command** job combines **code**, **command**, **environment**, **compute**, inputs and outputs. Bind an input into a script argument with **${{inputs.training_data}}** and parse the argument in the script. Submit with **ml_client.jobs.create_or_update()**.
- Use **ro_mount** to expose input files on demand or **download** to stage them locally. Folder outputs can use **rw_mount** or **upload** as supported. Declare an **Output** rather than assuming a local path becomes a shared artifact.
- An **Environment** defines the container image and dependencies. **image** selects a base image; **conda_file** adds packages; **BuildContext** supplies Docker build files. Create a new asset version when changing the execution environment.
- **Curated environments** offer prepared framework stacks; inspect included packages. Pin compatible dependency versions for reproducibility. In SDK v1, **ScriptRunConfig** packages source directory, script, arguments, environment and target for **Experiment.submit()**.
- For distributed PyTorch, **DistributedDataParallel** coordinates replicas and **DistributedSampler** shards examples across processes. Match process and GPU counts, preserve checkpoints, and avoid duplicate processing. **Batch normalization** uses learned running statistics during inference.

Track experiments with MLflow

- For external training, point **mlflow.set_tracking_uri()** at the workspace tracking URI, select **set_experiment()**, then **start_run()**. Azure ML jobs configure their tracking context; authenticate the external client separately.
- **log_metric()** records a numeric value; **log_metrics()** records a name-to-number mapping. Add a **step** for epoch histories. **log_param()** records configuration. **MlflowClient.log_batch()** batches tracking records; console prints are not tracked metrics.
- Use **log_artifact()** for a file, **log_artifacts()** for a directory, **log_text()** for text, and **log_dict()** for structured content. Log figures/images and tables as artifacts; a dictionary of arbitrary objects is not a numeric metric mapping.
- **mlflow.autolog()** captures supported framework metrics, parameters and models. **log_models=False** suppresses model logging; configure it when model artifacts are required. Save the model with its framework flavor and retain environment and input signature metadata.
- In SDK v1, **Run.log/log_list/log_table/log_image** track different value types. **get_metrics()** retrieves metrics; **get_file_names()** lists artifacts; **get_all_logs()** or **download_files()** retrieves logs/files. Use studio or job logs to investigate failures.

Training / Evaluation & Ethics

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

REPRODUCIBLE TRAINING

Explore Profile data → **Prepare** Fit transforms → **Train** Run jobs → **Track** Log results

Automated ML and forecasting

- Configure the **task**, **training_data**, **target_column_name**, compute and suitable **primary_metric**. Supply validation data or cross-validation settings. SDK v1 uses **AutoMLConfig** and **label_column_name**; **get_output()** returns the best run and fitted model.
- **Automatic featurization** can impute missing values and encode categories; **FeaturizationMode** controls the v2 setting. Algorithm allow/block lists restrict candidate learners. Disabling deep learning or featurization does not restrict training to linear models.
- For **forecasting**, identify the time column, target, series identifiers and frequency. Express the **forecast horizon in time steps**: seven days of twice-daily observations requires fourteen steps. Preserve temporal ordering during validation.
- Trial-count limits cap candidate exploration; concurrency and timeout control different resources. Classification optimizes classification metrics, regression numeric error metrics, and image tasks their task-specific scores. Review validation results before selecting a model.
- For image AutoML, **JSONL** records associate image locations with labels and are referenced through the required tabular input. NLP tasks distinguish single-label/multilabel **text classification** from **named entity recognition**, which labels token spans; configure the language with training settings.

03 Evaluate, tune and apply responsible AI

Classification metrics and thresholds

- A **confusion matrix** counts **TP**, **TN**, **FP** and **FN**. A false positive raises a false alarm; a false negative misses a positive case. Verify the chart's actual/predicted axis convention.
- **Precision = TP/(TP+FP)** measures reliability of positive predictions. **Recall = TP/(TP+FN)** measures positive-case coverage. **F1 = 2PR/(P+R)** balances precision and recall. Accuracy alone can conceal failure on rare classes.
- **ROC** plots true-positive rate against false-positive rate across thresholds; **AUC** summarizes ranking. Precision-recall curves are useful for rare positives. Choose the decision threshold for error costs; lowering it usually increases recall and false positives.
- **Macro averaging** gives each class equal weight; **weighted averaging** uses class support; **micro averaging** pools counts. Review per-class results alongside aggregates. **Weighted kappa** measures chance-adjusted agreement with penalties for ordinal disagreements.

Evaluation & Ethics

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

ASSESS THE CANDIDATE

- **Metrics** — Measure
- **Tuning** — Optimize
- **Explanations** — Understand
- **Fairness** — Compare groups

Regression metrics and model fit

- **MAE** averages absolute errors; **RMSE** penalizes large errors more strongly. Lower is better in target units. **R-squared** compares squared error with a mean-prediction baseline; one is perfect, zero matches that baseline, and negative values are possible.
- Inspect **residuals** for bias, changing variance and patterns; an actual-versus-predicted plot should approach the diagonal. Compare models on the same held-out data and metric. Designer follows **Train Model** → **Score Model** → **Evaluate Model**.
- **Overfitting** shows strong training performance with weaker validation results; **underfitting** performs poorly on both. Deeper trees generally lower bias and raise variance. Select complexity using validation performance rather than the smallest training error.
- **L1** penalizes absolute weights and can induce sparsity; **L2** penalizes squared weights. **Dropout**, training-data augmentation and early stopping can improve neural-network generalization. Cross-entropy is a classification **loss**, distinct from an activation function.

Hyperparameter search

- A **sweep job** varies training settings such as learning rate, batch size and tree depth. Log the metric with the exact **primary_metric** name. **Maximize** accuracy/AUC; **minimize** validation loss/error. SDK v1 uses **HyperDriveConfig**.
- **Grid** enumerates discrete **Choice** combinations; three values by two values gives six trials. **Random** supports discrete and continuous spaces. A **Sobol** rule improves space filling. **Bayesian** search uses completed trials; lower concurrency can improve its feedback.
- **Uniform** samples an interval, **Normal** a mean/standard-deviation distribution, and **LogUniform** samples uniformly in log space. Quantized distributions yield stepped values. Use a search space supported by the selected sampler.
- **max_total_trials** caps the trial count; **max_concurrent_trials** caps simultaneous trials; timeout limits duration. More parallelism needs more compute. In SDK v1, corresponding budget names include **max_total_runs** and **max_concurrent_runs**.

Evaluation & Ethics

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

ASSESS THE CANDIDATE

- **Metrics** — Measure
- **Tuning** — Optimize
- **Explanations** — Understand
- **Fairness** — Compare groups

Early termination policies

- **BanditPolicy** compares performance with the best run. **slack_amount** is an absolute allowance; **slack_factor** is relative. For a maximized metric, the relative cutoff is **best/(1+slack_factor)**, not best multiplied by one minus slack.
- **MedianStoppingPolicy** compares a run against the median of running-average performance across runs. **TruncationSelectionPolicy** cancels a configured percentage of the poorest performers. These use different references from Bandit's best-run comparison.
- **evaluation_interval** counts metric reports between evaluations; **delay_evaluation** gives trials a warm-up period. Report comparable validation metrics regularly so the policy can evaluate progress fairly.
- No early termination policy lets trials finish within their limits. SDK v1 **Bayesian HyperDrive** does not support an early termination policy. Use a compatible sampler/policy combination; aggressive termination can discard slow-starting configurations.

Explainability and Responsible AI

- **Global explanations** summarize overall feature influence; **local explanations** explain individual predictions. **TabularExplainer** supports tabular models; text/image explainers match their modalities. SHAP attributes contributions; a **MimicExplainer** uses an interpretable surrogate.
- **Permutation Feature Importance** shuffles one feature and measures metric degradation using a trained model and evaluation data. It describes global model reliance, not an individual prediction or causation; correlated features can obscure importance.
- The **Responsible AI dashboard** combines error analysis, explanations, fairness, counterfactuals and causal analysis for supported model/data configurations. Register a compatible model and supply the required data; MLflow formatting alone does not guarantee every dashboard feature.
- **Error analysis** finds weak cohorts. **Counterfactual what-if** asks which feasible feature changes could alter a prediction. **Causal analysis** studies interventions under causal assumptions. Feature importance alone cannot establish causal effects or regulatory compliance.

Evaluation & Ethics / Deployment & MLOps

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

ASSESS THE CANDIDATE

- **Metrics** — Measure
- **Tuning** — Optimize
- **Explanations** — Understand
- **Fairness** — Compare groups

Fairness and differential privacy

- Use **Fairlearn MetricFrame** to compare performance and selection rates across **sensitive groups**. **Demographic parity** concerns selection rates; **equalized odds** concerns both true-positive and false-positive rates; **equal opportunity** concerns true-positive rates.
- **ThresholdOptimizer** postprocesses a fitted classifier using group-specific thresholds. Fit the postprocessor with appropriate labeled calibration data and **prefit=True** to reuse the base estimator; sensitive features are needed for prediction. **ExponentiatedGradient/GridSearch** reductions retrain weighted estimators.
- For regression, compare group error or loss. Fairness constraints can trade accuracy against disparity. Removing a sensitive feature does not remove correlated proxies; select metrics according to the harm and decision context.
- **Differential privacy** limits how much one individual's data changes released aggregates by adding calibrated noise. **SmartNoise** supports private statistical analysis. Smaller positive **epsilon** means stronger privacy, usually with less utility; **delta** bounds an allowed relaxation of the privacy guarantee.
- Repeated private queries consume a **privacy budget** through composition; two epsilon-0.5 queries use epsilon 1 under basic sequential accounting. Bound data and contributions to control sensitivity. There is no universal epsilon threshold that guarantees safety or zero accuracy loss.

04 Deploy, monitor and automate models

Model assets and custom MLflow flavors

- Register model artifacts as named, versioned **model assets**; retain earlier versions for comparison and rollback. **Tags** hold editable metadata. In SDK v1, model properties have different mutability rules from tags.
- A v2 job output can be referenced through **azureml://jobs/<job>/outputs/<output>/paths/<path>**. An MLflow run artifact uses a **runs:/** URI. Select **mlflow_model** for a complete MLflow model directory, not merely a serialized weight file.
- An **MLflow flavor** describes loading and prediction. **python_function/pyfunc** supplies a common predict interface. A custom **PythonModel** can implement **load_context()** and **predict()**; a loader module can load external model data when ordinary pickling is unsuitable.
- Package required **artifacts**, source/library code, dependency definitions and input/output **signatures**. A file extension or framework tag alone does not install runtime dependencies. Compare new validation metrics explicitly before promotion; a higher version is not automatically a better model.

Deployment & MLOps

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

OPERATE THE MODEL

Register Version model → **Deploy** Online or batch → **Monitor** Quality and drift → **Retrain** Evaluate again

Pipeline components and data flow

- A **component** defines reusable code, environment, inputs and outputs; a **pipeline job** connects components. Register custom component YAML for designer use. A registry can share assets across workspaces; workspace assets remain locally scoped.
- Bind an upstream **output** to the downstream **input** to establish a dependency. Independent branches can run in parallel; list order alone does not guarantee sequencing. Use CPU preprocessing and GPU training where each is cost-effective.
- SDK v1 uses **PythonScriptStep**, **PipelineData** and **OutputFileDatasetConfig**. **as_input()** binds an output downstream; **register_on_complete()** can register a resulting dataset. Shared datastore outputs survive beyond a compute node's local filesystem.
- Reuse assumes unchanged inputs and deterministic behavior. In SDK v1, **allow_reuse=False** disables step reuse; **regenerate_outputs=True** forces a run. Mutable storage can evade change detection. In v2, configure component determinism and pipeline **force_rerun** deliberately.

Online endpoints and scoring code

- An **online endpoint** accepts requests and returns synchronous predictions. A **deployment** supplies the model, environment, scoring code and compute resources. **Managed online endpoints** manage serving infrastructure; Kubernetes endpoints use configured cluster capacity.
- In a custom scoring script, **init()** loads the model once per worker and **run(data)** parses requests, applies preprocessing, predicts and serializes results. SDK v2 **CodeConfiguration** selects the code folder and **scoring_script**.
- An eligible **MLflow model** supports no-code deployment with generated scoring and environment configuration. Supply custom code/environment when changing inference behavior. For restricted egress, prepare required packages in an accessible image rather than depending on public downloads at startup.
- Inference inputs contain **features**, without requiring the unknown target. Reuse fitted preprocessing and reverse target transformations where needed: a log-price model requires exponentiation to return prices. In designer, validate the inference pipeline and its input/output schema before deployment.
- Provision **GPU inference** only when the serving workload benefits and the environment supports it. Historical SDK v1 patterns use **LocalWebservice** or **ACI** for testing and **AKS** for scalable serving; an AmlCompute training cluster is a batch/job target.

Deployment & MLOps

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

OPERATE THE MODEL

Register Version model → **Deploy** Online or batch → **Monitor** Quality and drift → **Retrain** Evaluate again

Endpoint access, traffic and release

- **auth_mode='key'** uses a key that remains valid until rotated. **aml_token** uses expiring Azure ML tokens. **aad_token** uses Microsoft Entra tokens for managed online endpoints; Kubernetes endpoints support key or aml_token modes.
- Separate **client identity** permissions from the **endpoint identity** used to read models, images or storage. TLS protects transport; it does not select an authentication method. SDK v1 **auth_enabled** and **token_auth_enabled** are separate configuration names.
- Consumers need the **scoring URI**, authentication material and request schema. Test JSON inputs, output types and error handling. Route a small traffic percentage to a new deployment, compare behavior and retain a tested rollback path.
- **Replica autoscaling** addresses serving demand; the backing cluster also needs adequate nodes. Monitor latency, throughput and saturation. Track endpoint/deployment costs, and gate automated release on evaluation rather than simply deploying every newly registered version.

Batch inference and parallel scoring

- A **batch endpoint** submits asynchronous jobs over large datasets and returns job information to poll. A model **batch deployment** references the registered model and an **AmlCompute cluster**. Jobs can scale cluster nodes down after completion.
- MLflow batch deployments can generate scoring behavior; use a custom **init()** and **run(mini_batch)** script to customize processing. Invoke the default deployment or name a deployment explicitly; inspect the resulting job and output artifacts.
- **mini_batch_size** controls work per scoring call; **instance_count/node_count** controls nodes; **max_concurrency_per_instance/process_count_per_node** controls per-node parallelism. Retry, timeout and error-threshold settings govern failures, not batch size.
- SDK v1 **ParallelRunStep** uses **ParallelRunConfig** on AmlCompute. For FileDataset, mini-batch size counts files. **append_row** aggregates returned rows into **parallel_run_step.txt**; **summary_only** leaves output writing to the script.

Deployment & MLOps

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

OPERATE THE MODEL

Register Version model → **Deploy** Online or batch → **Monitor** Quality and drift → **Retrain** Evaluate again

Debug and monitor deployments

- For SDK v2, use **ml_client.online_deployments.get_logs()** with endpoint and deployment names. **inference-server** logs contain scoring print/log statements; **storage-initializer** logs help diagnose model download and initialization issues.
- SDK v1 **service.get_logs()** exposes deployment/runtime logs. Reproduce scoring failures with local Docker and a **LocalWebservice**; **reload()** supports iterative scoring-script changes. Check model paths, missing dependencies, schema mismatches and memory before resizing blindly.
- Use **Azure Monitor**, **Log Analytics** and **Application Insights** for request rates, failures, latency and resource signals. Configure input/output data collection deliberately for data monitoring; ordinary telemetry does not automatically retain every payload.
- Protect collected inference data, limit sensitive fields and retain it according to policy. Track data quality and model outcomes alongside system availability; healthy containers can still serve inaccurate predictions.

Drift detection and retraining

- **Data drift** changes input distributions; **concept drift** changes the input-to-target relationship. Compare representative production features with a training/reference baseline. A detected shift is a signal to investigate, not proof of reduced predictive accuracy.
- Keep timestamps in the target production dataset to analyze windows. Monitor missing values, ranges, categories and distribution changes. Measure predictive performance when ground-truth labels arrive; prediction values alone are not ground truth.
- SDK v1 **DataDriftDetector.create_from_datasets()** defines baseline/target monitoring; **backfill()** evaluates a historical date range. Target time-series configuration and selected features matter more than matching a predicted-label column.
- A retraining workflow should validate new data, train, compare evaluation results, register an acceptable candidate and release through a controlled deployment step. Preserve data/model lineage and check performance by cohort after release.

Deployment & MLOps

Designing and Implementing a Data Science Solution on Azure

KNOWLEDGE CHEATSHEET

OPERATE THE MODEL

Register Version model → **Deploy** Online or batch → **Monitor** Quality and drift → **Retrain** Evaluate again

Schedules and event-driven automation

- A v2 **RecurrenceTrigger** uses **frequency** and **interval**, with optional start time and time zone. A **JobSchedule** associates the trigger with **create_job**. SDK v1 schedules instead reference a published pipeline and **ScheduleRecurrence**.
- A published batch pipeline's REST call starts a run and returns an identifier; it does not wait for final predictions. Parameterize inputs so callers can select datasets without duplicating the pipeline.
- **Event Grid** routes discrete events such as Blob creation or supported ML lifecycle events. **Logic Apps** provides connector workflows and notifications; **Functions** executes custom handlers; **Data Factory** orchestrates data refresh and processing.
- Historical Azure ML event terms include **RunCompleted** and **DatasetDriftDetected**. Filter for the actual event and connect it to an authorized handler. Azure DevOps pipelines use service connections, model artifacts and release gates for automated deployment.